Java in Hindi

BccFalna.com
097994-55505

Kuldeep Chand

This EBook is not Just **Core Java**, but also includes some concepts of Advance Java like Basics of JDBC, **Event Driven Programming**, GUI development with AWT and Basics of Java Networking too.

In Java, all GUI development like *SWT/Swing, JavaFX* etc… are totally based on **AWT**. So, learning AWT helps very much in learning GUI Development using Java. So, in this EBook, I have covered GUI Development from and covered **AWT and Event Driven Programming** with Good Detail in Last Chapter. So that, after reading this EBook, you can start developing GUI Applications using Java easily.

Even Applets are out of market now, but I have included it frequently in this EBook to easily using and understanding GUI Development. Basics.

I have covered each Java Programming Concept with hundreds of example programs. So, it would be very easy to learn Java with this EBook.

Java

In Hindi

□ □ □
Kuldeep Chand

BetaLab Computer Center
Falna

Programming Language JAVA in Hindi

Lead Editors: Kuldeep Chand

Distributed to the book trade worldwide by Betalab Computer Center, Behind of Vidhya Jyoti School, Falna Station Dist. Pali (Raj.) Pin 306116

e-mail bccfalna@gmail.com

or

visit http://www.bccfalna.com

For information on translations, please contact Betalab Computer Center, Behind of Vidhya Jyoti School, Falna Station Dist. Pali (Raj.) Pin 306116

Phone 97994-55505

This book is dedicated to those

who really wants to be

a

PROFESSIONAL DEVELOPER

INDEX OF CONTENTS

Contents

JAVA

BASICS AND PROGRAMMING FUNDAMENTALS

Java – Basics and Programming Fundamentals

आज हम देख सकते हैं Internet व Mobiles का कितना विस्तार हो चुका है। आज Internet इतना बढ चुका है कि दुनिया की जो भी जानकारी चाहिए, Internet पर उस जानकारी को प्राप्त किया जा सकता है। आज इस Internet की वजह से दुनिया बिल्कुल छोटी सी हो गई है। हम जब चाहें जिससे चाहें बात कर सकते हैं या Online Meeting कर सकते हैं। दुनिया की लगभग सारी चीजें आज Internet से जुडी हुई हैं। Internet पर आज हम केवल Texts ही नहीं पूरे Multimedia को देखते हैं, जिसमें Sound, Video, Animation, Graphics आदि जो कुछ भी हो सकता है, सब है।

लेकिन आज हम जिस तरह का Internet देख पा रहे हैं, कुछ समय पहले तक Internet ऐसा नहीं था। Multimedia की विभिन्न चीजों को Internet पर सम्भव बनाने में Java का बहुत बडा सहयोग रहा है। वास्तव में Java का विकास केवल Internet के लिए किया गया था, लेकिन आज इसका प्रयोग केवल Internet के WebPages बनाने के लिए ही नहीं होता है, बल्कि आज ये बडे–बडे Standalone Application Software व Distributed Application Develop करने की सबसे आसान व उपयोगी भाषा है। जितनी आसानी से हम Java का प्रयोग करके एक Distributed System Create कर सकते हैं, उतनी आसानी से किसी भी अन्य भाषा का प्रयोग करके हम Internet के लिए बडे Software Develop नहीं कर सकते हैं।

आपने भी लोगों को ये कहते सुना होगा कि Computer Programming काफी कठिन काम है। ये हर किसी के बस की बात नहीं है। Computer Programmer बनने के लिए MCA, B-Level जैसे Degree Level Courses और हजारों रूपए के Hardware व Software की जरूरत होती है।

साथ ही वही Programmer बन सकता है जिसका दिमाग Computer की तरह काम करता हो यानी बहुत तेज हो। जो घण्टों किसी समस्या का समाधान प्राप्त करने के लिए धैर्यपूर्वक बैठ सकता हो। आदि–आदि।

एक अच्छा Programmer बनने के लिए ये सभी बातें जरूरी होती थीं लेकिन तब, जब Programmer किसी Assembly Language या Cobol, Pascal आदि जैसी किसी Language में Programming करना सीखता था। Java के साथ इससे बिल्कुल उल्टा है।

Java में Programming सीखना जितना आसान है, उनता शायद ही किसी Language को सीखना हो। इसमें बस कुछ Basic Concepts ध्यान हों, तो बहुत ही आसानी से कोई भी आवश्यकतानुसार Program बना सकता है और उसे Use कर सकता है। साथ ही वह अपने Application को Internet पर भी उतनी ही आसानी से चला सकता है जितना अपने स्वयं के Computer पर।

हम Programming को इतना Hard इसलिए मानते हैं क्योंकि ऐसा हमें अन्य Programmers ने कहा है। ये Programmers की Monopoly है ताकि उन्हें अच्छी Payment प्राप्त हो सके। यदि सभी लोग ऐसा कहने लगें, कि Programming बहुत ही सरल काम है, तो क्या Programmers को किसी Program के लिए उतने पैसे मिलेंगे जितने आज मिल रहे हैं।

शायद नहीं, इसीलिए सभी Programmers कहते हैं कि Programming सबसे कठिन काम है। हमारे देश में लोग Computer Programming को इसलिए कठिन समझते हैं, क्योंकि उन्हें उनकी भाषा में लिखे हुए Matter प्राप्त नहीं होते। दूसरी बात ये कि Computer को ठीक से तभी समझा जा सकता है, जब English पर अच्छी पकड हो। लेकिन ऐसा जरूरी नहीं है। Computer Programmer बनने के लिए अच्छी English उतनी जरूरी नहीं है जितनी तथ्यों को समझने व समझाने की योग्यता की जरूरत है।

Programming सीखने के लिए सबसे पहली चीज ये तय करनी होती है कि आखिर किस Language से Programming की शुरूआत की जाए। हालांकि सभी Languages में लगभग कुछ तथ्य समान ही होते हैं। जैसे Data Types, Operators, Conditional and Looping Statements आदि, लगभग सभी Languages में थोडे बहुत अन्तर के अलावा समान ही होते हैं और उन्हें Use करने का तरीका कभी काफी हद तक सभी Languages में समान होता है।

यदि आपने "C" Language में या "C++" Language में थोडी बहुत Programming की है और Programming के Basic Concepts आपको Clear हैं, तो Java आपके लिए आगे बढाने वाली सबसे अच्छी भाषा हो सकती है। हालांकि हर Programming Language की अपनी कुछ अलग विशेषता होती है जिसके आधार पर अलग–अलग Requirement के आधार पर अलग–अलग भाषा अधिक उपयोगी होती है। कुछ काम ऐसे भी होते हैं जो किसी Language में आसानी से Perform होते हैं और कुछ Languages में किसी भी तरह से उन कामों को नहीं किया जा सकता है।

उदाहरण के लिए यदि Fastly कोई GUI Application Software Develop करना हो, तो Microsoft Company का Visual Basic सबसे सरल Programming Language है। इसमें आज हजारों Software बन चुके हैं जिनका प्रयोग Personnel Use व Business Use दोनों स्थानों पर बहुत होता है। लेकिन Visual Basic Programs की कमी ये है कि इनकी Speed किसी अन्य Languages जैसे कि Borland C++ या Visual C++ में लिखे गए Programs की तुलना में कम होती है। इस Speed की कमी को तब महसूस किया जा सकता है, जब Program में बहुत सारे Graphics का प्रयोग किया गया हो।

जैसे कि यदि Visual Basic में Screen Saver या कोई Game Develop किया जाए तो इनकी Speed काफी कम होती है। इसलिए जो Professional Programmers होते हैं वे कभी भी Graphics Programming के लिए Visual Basic का प्रयोग नहीं करते हैं।

हालांकि Java को Visual Basic की तुलना में सीखना काफी कठिन है, लेकिन फिर भी Java को सीखना कई मायनों में काफी उपयोगी साबित होता है। Java की सबसे बडी विशेषता तो यही है, कि इसमें Develop किए गए Programs को हम World Wide Web पर Use कर सकते हैं। यदि आपने Internet Surfing की है तो आपने विभिन्न Websites पर कई Animations, Sounds आदि देखें व सुने होंगे। ये सभी काम Java में काफी आसानी से किए जा सकते हैं। यानी यदि आप

कोई ऐसा Program बनाना चाहते हैं, जिसको Internet पर भी चलाया जा सकता है, जैसे कि Online Games, तो आपको Java की जरूरत होगी।

Java की दूसरी विशेषता ये है कि Java का Program एक विशेष तरीके से लिखा जाता है जिसमें हमें Java के सभी नियमों का पूरी तरह से पालन करना पडता है। यदि हम Java के किसी छोटे से नियम को भी Neglect करते हैं, तो एक छोटे से "Hello World" Program को Create करके Compile करने में भी हमें काफी परेशानियों व Errors का सामना करना पड सकता है।

Java को ऐसा इसलिए बनाया गया है ताकि जितनी भी Errors व परेशानियां आनी हैं, वे Program के Creation के समय ही आ जाएं, ताकि जब Program पूरी तरह से तैयार हो जाए, तब किसी प्रकार की परेशानी ना आए और Program Reliable, उपयोगी व Error Free हो। और वास्तव में जावा के Programs अन्य Languages की तुलना में काफी ज्यादा Reliable होते हैं।

कई अन्य Languages जैसे कि Visual Basic आदि में Program शुरू से अन्त तक कोई Error नहीं देता लेकिन किसी ना किसी जगह पर ऐसी Error Generate करता है, जिसका Solution करने में हमें उतना समय लग जाता है जितना उस Program को Create करने में नहीं लगता।

Java का विकास Sun Microsystems के एक Developer James Gosling ने किया था। उन्हें इसका विकास करने की जरूरत इसलिए पडी क्योंकि वे "C++" Language का प्रयोग करके एक Project बना रहे थे लेकिन उन्हें वह परिणाम प्राप्त नहीं हो पा रहा था जो वे चाहते थे। इसलिए उन्होंने स्वयं एक Language Develop की जिससे उनकी Requirement पूरी हो सके। इसी Language का नाम **"Java"** है।

Java को सीखना किसी भी अन्य Language को सीखने की तुलना में अधिक सरल है। ज्यादातर Languages एक दूसरे के लगभग समान ही हैं। इसलिए यदि एक Language में Mastery कर ली जाए तो बाकी की अन्य Languages में किसी Programmer को ज्यादा परेशानी नहीं आती है। वह आसानी से किसी भी Language में पकड बना लेता है। लेकिन इसके लिए जरूरी है कि उसे कम से कम एक Language में काफी जानकारी हो।

जो लोग पहले "C" या "C++" या दोनों सीख चुके हैं उन्हें Java को सीखने में कोई परेशानी नहीं आती है बल्कि वे उन लोगों की तुलना में ज्यादा जल्दी से Java को सीख लेते हैं और Java पर पकड बना लेते हैं, जिन्होंने "C" या "C++" नहीं सीखी है। अगर हम ऐसा कहें कि Java "C" व "C++" का मिलाजुला रूप है और Java में से उन चीजों को हटा दिया गया है, जिनको "C" व "C++" Language में सीखने में परेशानी आती थी, तो गलत नहीं होगा।

लेकिन इसका मतलब ये नहीं है कि Java को सीखने से पहले "C" व "C++" को सीखना जरूरी है। हालांकि यदि पहले "C" व "C++" सीखा जाए तो Java को समझना व सीखना सरल होता है लेकिन फिर भी हम Java से Programming सीखना शुरू कर सकते हैं। ये अपने आप में ही एक पूर्ण Language है। Java सीखने के बाद भी किसी भी अन्य Language को उतनी ही आसानी से

सीखा जा सकता है जितनी आसानी से किसी और Language को सीखने के बाद Java को सीखा जाता है।

High Level Programming Languages के विकास की यदि बात करें, तो UNIX Operating System के लिए एक भाषा का विकास किया गया था, जिसका नाम "C" Language दिया गया। इस भाषा का विकास मुख्य रूप से Operating System UNIX को Develop करने के लिए किया गया था। UNIX Operating System Develop हो जाने के बाद UNIX Operating System के लिए Applications Software को Develop किया जाने लगा।

चूंकि "B" Language का विकास एक System Software को Develop करने के लिए किया गया था, इसलिए विभिन्न Programmers को इस Language में UNIX के लिए Application Software लिखने में परेशानी आती थी। इसलिए इस "B" Language को और सरल बनाया गया ताकि Programmers इस Language में Application Programs Develop कर सकें। इस Developed Language को "C" Language नाम दिया गया।

"C" Language शुरूआत में काफी उपयोगी साबित हुई लेकिन जिस तरह से हर चीज में विकास होता है, उसी तरह से Computer Technology में भी विकास हुआ। धीरे–धीरे Application Software इतने बडे व जटिल होने लगे, कि "C" Language में Develop किए गए Programs को Manage व Maintain करना काफी कठिन हो गया। साथ ही जैसे–जैसे समय बीतता गया, Software की जटिलता भी बढती गई।

इसलिए एक बार फिर Programmers को ये महसूस होने लगा कि उन्हें कुछ और अधिक सरल तरीके की जरूरत है, जिससे वे बडे व जटिल Programs को Handle कर सकें। ये नया तरीका भी जरूरत के अनुसार Develop किया गया और इस तरीके को Object Oriented Concept कहा गया। इस Object Oriented Concept को ध्यान में रख कर Programming language "C" में फिर विकास किया गया और इस विकास का परिणाम "C++" Programming Language के रूप में प्राप्त हुआ।

हालांकि आज Java का जिस उद्देश्य के लिए ज्यादातर प्रयोग किया जा रहा है और जावा जिस प्रकार की Programming के लिए जानी जाती है, वास्तव में Java का विकास इसके लिए नहीं किया गया था। जावा का विकास General Electronic Equipments को अधिक समझदार बनाने के लिए किया जा रहा था, ताकि विभिन्न प्रकार के Equipments को Artificial Intelligence प्रदान की जा सके। हालांकि ऐसा तो नहीं हो सका, लेकिन जावा एक Dynamic Internet Programming Language के रूप में काम आने लगी।

Java का विकास करने वाले लोग जिस Project पर काम कर रहे थे, वे उसमें "C++" Language का प्रयोग कर रहे थे, जो कि "C" Language का ही विकसित रूप है। लेकिन वे जो करना चाहते थे, वैसा "C++" के प्रयोग से नहीं कर पा रहे थे। इसलिए उन्होंने एक नई Language Develop की। इस Language को उन्होंने "C" व "C++" के आधार पर ही Develop किया है। वे Java को

एक बहुत ही सरल Language बनाना चाहते थे, इसलिए उन्होंने "C" व "C++" की सभी आसान Concepts को ज्यों का त्यों उपयोग में लिया और जटिल Concepts को छोड दिया।

उन्होंने Java Language के Programming Syntax को भी लगभग वैसे ही उपयोग में लिया जिस तरह से "C" व "C++" में लिया जाता है। साथ ही उन्होंने कई अन्य Languages के Concepts का भी प्रयोग जावा में किया ताकि इसमें किसी भी प्रकार का Software आसानी से बनाया जा सके और Software पूरी तरह से विश्वसनीय बने।

इस तरह से Java केवल "C" व "C++" का Modified Version ही नहीं है, बल्कि कई अन्य Languages के Concepts पर आधारित एक पूर्ण Programming Language है। हालांकि इसके ज्यादातर Syntax व Coding Procedures "C" व "C++" Language के अनुसार हैं, इसलिए इसे "C++" Language का Modified Version भी कहा जा सकता है।

जैसे–जैसे जरूरत बढती जाती है और जरूरत का स्वरूप बदलता जाता है, वैसे–वैसे Programming Languages को भी Develop करना जरूरी हो जाता है, ताकि वर्तमान की विभिन्न जरूरतों को पूरा किया जा सके। इसी तथ्य पर अब जावा से आगे की Language को Microsoft Company ने Develop किया है। इस Language का नाम "C#" (**CSharp**) है। इस Language में "C", "C++" व Java तीनों Languages की विभिन्न विशेषताओं को Include किया गया है। Microsoft इस Language में Software Development के लिए पूरा IDE प्रदान करता है, जिसमें आज की जरूरत के अनुसार विभिन्न कामों को किया जा सकता है।

लेकिन इसका मतलब ये नहीं है कि Java अब पुरानी हो चुकी है। आज भी Java का Market में अपना एक अलग व महत्वपूर्ण स्थान है और Java को सीखे व समझे बिना, अगली Generation की Languages को समझना काफी मुश्किल है।

हालांकि जावा का विकास जिस काम के लिए किया जा रहा था, उस काम के लिए जावा उपयोगी नहीं बन पाया। लेकिन जब जावा के Developers ने देखा कि इस Language का प्रयोग Internet की Interactive Programming में काफी उपयोगी साबित हो सकता है, तब उन्होंने इस Language को Internet के लिए Develop करना शुरू किया। वे जिस **Platform Independent Equipment Technology** पर काम कर रहे थे, वह तकनीक Internet के लिए उपयोगी साबित हो गई।

Features of JAVA

Java केवल एक Programming Language ही नहीं है बल्कि ये एक Platform भी है। जब Sun Microsystems ने November 1995 में Java को दुनिया से परिचित करवाया तब Company के Cofounder Bill Joy ने Java की निम्न परिभाषा दी थी कि

Java एक Small, Simple, Safe, Object-Oriented, Interpreted या Dynamically Optimized, Byte-Coded, Architecture-Neutral, Garbage-Collected, Multithreaded Programming Language है जिसमें Distributed, Dynamically Extensible Programs लिखने के लिए एक Strongly Typed Exception-Handling Mechanism है। जावा के इन्हीं गुणों को जावा के **Features** भी कहते हैं।

Small and Simple

Java एक छोटी और सरल भाषा है जिसे आसानी से सीखा जा सकता है। जावा को इस तरह से Design किया गया है कि इसे कोई भी Programmer आसानी से सीख सके और Computer Programming के Internal Functionality को जाने बिना भी ज्यादा से ज्यादा Efficient Program Develop कर सके। यदि किसी Programmer को किसी भी Programming Language का थोडा भी ज्ञान है, तो वह बहुत ही आसानी से व जल्दी से Window Based Application व Internet Based Distributed Application (Applets) Develop करना सीख सकता है।

जब जावा को पहली बार Release किया गया था, तब वह काफी छोटी भाषा थी। लेकिन आज ये काफी बडी भाषा बन चुकी है और सभी प्रकार के Applications को Efficiently Develop करने में सक्षम है। ये Language C/C++, Simula, Ada जैसी कई अन्य Languages से प्रेरित है, लेकिन इसकी ज्यादातर Coding C++ Language के समान ही है। इसलिए किसी C/C++ Programmer को जावा सीखने में कोई कठिनाई नहीं होती है।

इस Language में C व C++ के अच्छे Features को Use कर लिया गया है जबकि इन Languages के Confusing तथा Typical Features को छोड कर उनके स्थान पर अधिक सरल Concepts को Include कर लिया गया है। जैसे C++ के Operator Overloading व Pointer जैसे Concepts को जावा में छोड दिया गया है, जबकि Multithreading जैसी Advance Technique को Add कर लिया गया है।

Object Oriented

Java में हर चीज Object व Class के रूप में परिभाषित है, जिसे Object Oriented Programming Concept कहा जाता है। OOPS हमें Abstraction and Encapsulation, Polymorphism और Inheritance जैसे Features प्रदान करता है, जिससे हम एक समस्या को उसी तरह से Computer में Logically Organize कर सकते हैं, जिस तरह से समस्या Real World में Actually या Physically Organized रहती है। जावा में बहुत सारी जरूरी Classes पहले से ही Packages के Form में हमें प्राप्त होती है, जिन्हें बिना Rewrite किए हम ज्यों का त्यों Use कर सकते हैं।

Distributed

Java के Programs Network पर यानी Web Pages पर भी Execute होते हैं। इसलिए इसे Distributed Language कहा जाता है। Distribution का मतलब ये होता है कि Java के Program किसी भी Platform पर Run हो सकते हैं।

हम जानते हैं कि आज कई तरह के Operating Systems उपलब्ध हैं और अलग–अलग लोग अपनी जरूरत व इच्छा के अनुसार अलग–अलग Operating Systems का प्रयोग करते हैं। कोई Windows Operating System Use करता है तो कोई Linux तो कोई MacOS या OS/2 Use करता है। ये सभी अलग–अलग Platform कहलाते हैं।

यदि हम Windows Based Computer पर कोई Program "C" या Visual Basic जैसी भाषा में Create करते हैं, तो वे Program उन सभी Computers पर आसानी से Run होते हैं जो Windows को Use करते हैं।

लेकिन यदि इन्हीं Programs को Linux या MacOS पर Execute करने की कोशिश की जाए तो ये Program उस Operating System पर Execute नहीं होते। इन Platforms के लिए Program को वापस इन्हीं Platform वाले Computers पर Compile करना पडता है। जबकि Java के साथ ऐसा नहीं है।

जावा में हम किसी भी Platform पर Program Create करके Compile करें, वे Program सभी अन्य Platforms पर समान रूप से Execute होते हैं। यानी Java के Programs को विभिन्न Platforms पर Distribute किया जा सकता है। इसलिए Java को Distributed Language कहा जाता है।

जावा को इस प्रकार से Design किया गया है कि हम इसमें ऐसे Applications Develop कर सकें, जो Internet पर चल सके। इस Language में ये Ability है कि ये Data व Program दोनों को Internet पर विभिन्न Computers पर Share कर सकता है। जावा Applications Remote Objects को भी उतनी ही आसानी से Access व Open कर सकते हैं, जितनी आसानी से वे Local Computer के Objects को Open व Access करते हैं। जावा ऐसी Networking की सुविधा प्रदान करता है कि विभिन्न Remote Locations पर स्थित विभिन्न Programmers एक ही Single Project पर समान समय पर एक साथ काम कर सकते हैं।

Compiled and Interpreted

ज्यादातर अन्य Languages के Programs या तो Compile होते हैं या फिर Interpreted होते हैं। लेकिन Java के Programs Compile भी होते हैं और Interpreted भी। Java के Programs को सबसे पहले Compile किया जाता है। Java के Program Compile होने के

बाद सीधे ही Machine Language में Convert नहीं होते हैं, बल्कि ये Source Code व Machine Code के बीच की स्थिति में Convert होते हैं जिसे **Bytecodes** कहा जाता है।

इन Bytecodes को जब किसी भी Platform पर Run करना होता है तब ये Bytecodes उस Computer के Platform के अनुसार Interpreted हो कर पूरी तरह से उस Machine के अनुसार Machine Code में Convert होते हैं और उस Platform पर Execute हो सकते हैं।

Robust and Safe

Java के Programs में Errors आने की सम्भावना अन्य Languages की तुलना में बिल्कुल कम होती है। इसलिए Java के Programs को Robust कहा जाता है। इसके Compiler में विभिन्न प्रकार से Generate होने वाली Errors को Handle करने के लिए कई Built-In तरीके Develop कर दिए गए है और जावा को इस तरह से Design किया गया है, कि एक बार सही तरीके से Compiled Program में कभी भी Error आने की सम्भावना नहीं रहती है। जितनी भी Errors आनी होती हैं, वे सभी Program Development व Testing के समय ही आ जाती हैं, जिन्हें Handle कर लिया जाता है।

इसमें Compile Time व Runtime दोनों स्थानों पर विभिन्न प्रकार के Errors के लिए विभिन्न Data Types की Checking होती है। विभिन्न प्रकार के Objects द्वारा ली जाने वाली Memory को ये स्वयं ही Release कर देता है, जिससे हमें इस बात की चिन्ता करने की जरूरत नहीं होती है, कि हमने सभी Unrequited Objects को Destroy करके उनकी Memory को Release किया या नहीं।

जावा में Exception Handling के लिए भी सुविधा प्रदान की गई है, जिसका प्रयोग हम Serious Errors को Trap करने व उन्हें Solve करने के लिए कर सकते हैं, जिससे हमारे Program की और सुरक्षा हो जाती है।

जब हम Internet की बात करते हैं, तब Security काफी मायना रखती है। जावा स्वयं ही विभिन्न प्रकार के Memory Management व Memory Access से सम्बंधित काम करता है, इसलिए ये कभी भी Memory व उसमें Stored Data को गलत तरीके से Access करने की छूट नहीं देता है।

इस वजह से Applet द्वारा किसी Computer में Virus आने की सम्भावना ही नहीं होती है। क्योंकि जावा में Pointers की सुविधा नहीं है जो Directly Memory को Access कर सके, इसलिए हम किसी भी Computer की Memory को Directly Access नहीं कर सकते हैं। साथ ही जावा Applets कभी किसी Client Computer के Resources को Access नहीं करते हैं, इसलिए जावा Applets कभी भी Clients के Computer या उसके Data को नुकसान नहीं पहुंचा सकते हैं।

Architecture Neutral / Platform Independent / Portable / Byte Coded

Java के Bytecodes विभिन्न प्रकार के Processors व Operating Systems पर समान रूप से Run हो सकते हैं। इसलिए इसे Architecture Neutral or Portable कहा जाता है। जावा के Programs को केवल एक ही बार Develop करना होता है। एक बार इसे Develop करने के बाद इसे किसी भी Computer पर किसी भी Platform पर Run किया जा सकता है।

यदि Operating System, System Resources या Processor में Change किया जाता है, तब भी हमें जावा के Program में किसी प्रकार का Change करने की जरूरत नहीं होती है। यही जावा के सबसे ज्यादा Popular होने की मुख्य वजह है, जिससे हम जावा का प्रयोग Internet Programming के लिए करके World Wide Web पर Run होने वाले Applications Develop करते हैं और विभिन्न Computers को आपस में Interconnected करते हुए World Wide Web पर काम करने के लिए ऐसे Programs को Use करते हैं।

हम जावा Applet को Remote Computer से Download कर सकते हैं और फिर उसे अपने Computer पर Run कर सकते हैं। इस प्रकार की सुविधा होने से एक User को उसके घर पर ही विभिन्न प्रकार के Applications व Applets प्राप्त हो जाते हैं, जिन्हें वह Use करना चाहता है।

जावा दो तरीकों से Portable होता हैः एक तो जावा Compiler Byte Codes Instructions Generate करता है, जिसे किसी भी Computer पर Implement किया जा सकता है और दूसरा जावा के Primitive या Basic Data Types Machine पर निर्भर नहीं होते हैं बल्कि जावा Platform पर निर्भर होते हैं। यानी किसी भी Compute पर जावा के सभी Data Types की Size समान होती है, चाहे हम Pentium Computer पर जावा Program को Use करें, चाहे AMD पर।

Garbage Collective

Java एक Programmer को Memory Manage करने की सुविधा प्रदान नहीं करता है बल्कि जरूरत के अनुसार स्वयं ही Memory Management करता है। इसलिए Programmer द्वारा Memory Management के समय किसी दूसरे Data को नुकसान पहुंचाने की सम्भावना नहीं होती है। इसलिए ये Language “C” व “C++” जैसी भाषाओं की तुलना में अधिक सुरक्षित या Secure Language है।

High Performance

Java के Program Interpreted Mode में Run होते हैं लेकिन फिर भी अन्य Interpreted Based Languages की तुलना में Java की Speed व Performance बहुत अच्छी होती है। इसलिए इसे High Performance Language कहा जाता है।

Multithreaded and Interactive

Java ये सुविधा प्रदान करता है कि एक ही Software Program के विभिन्न भागों को एक ही समय में एक साथ Run किया जा सकता है। इसलिए इसे Multithreaded Language कहा जाता है।

उदाहरण के लिए मानलो कि हम किसी Program से Audio Sound तो सुन ही रहे हैं, साथ ही उसी Program में Scroll Bars को भी Use कर रहे हैं। किसी Window में एक तरफ कोई Movie Clip तो Play हो ही रहा है, साथ ही हम किसी अन्य Movie Clip को Open करने के लिए Open Dialog Box में किसी दुसरी Movie Clip को भी खोज रहे हैं। इस तरह से एक ही Program के विभिन्न हिस्सों का एक ही समय में एक साथ Run होना Multithreaded Concept के कारण ही सम्भव होता है।

Dynamic and Extensible

Java में एक ही Program के कई Versions को एक साथ Maintain किया जा सकता है। इसलिए इसे Dynamic Language भी कहा जाता है। यानी जावा एक Dynamic Language है। जावा में किसी Program के लिए Required Classes जावा के Program के Run होते समय उसमें Link हो सकती है और जैसे ही उस Class का काम समाप्त होता है, वह Class स्वयं ही Memory से Release हो जाती है। इस प्रक्रिया को Dynamic Process कहा जाता है।

जावा एक Query द्वारा ये भी Determine कर सकता है कि Program के Run Time में उससे कौनसी Class Link हो रही है। साथ ही वह Program के Run Time में भी किसी भी Dynamic Class या Dynamic Link Library से Link हो सकता है और Run Time सुविधाओं को प्राप्त कर सकता है। इस प्रक्रिया को Dynamic Linking भी कहते हैं।

जावा हमें अन्य Languages के Methods को भी जावा में Use करने की सुविधा प्रदान करता है। इन Methods को Native Methods कहते हैं और ये Program के Run Time में Dynamically Link हो कर अपना काम करते हैं। यानी हम जावा में अन्य Languages की सुविधाओं को Use करके जावा के Program की क्षमताओं को बढा सकते हैं या Extend कर सकते हैं। इसी प्रक्रिया को जावा की Extensibility कहते हैं।

Java के Programs कई प्रकार के होते हैं, उनमें से कुछ निम्नानुसार हैं :

Applications

ये ऐसे Programs होते हैं जिन्हें Execute होने के लिए किसी Browser की जरूरत नहीं होती है। ये Stand Alone होते हैं और किसी भी Computer पर Run हो सकते हैं। इन्हें Command Prompt पर Run किया जा सकता है।

Applets

ये ऐसे Programs होते हैं जिन्हें Run होने के लिए Browser की जरूरत होती है। ये Programs Web Pages पर Run होते हैं। एक Applet Program कभी भी किसी Local Machine के Resources को Access नहीं करता है।

Servlet

ये Programs Web Servers की Functionality को बढाने के लिए लिखे जाते हैं। सामान्यतः इनका कोई GUI नहीं होता है।

Packages

ये Java की Classes का एक Collection होता है जिसे किसी भी अन्य Java Program में आसानी से Reuse किया जा सकता है।

Object Oriented Concept Programming करने का एक असाधारण लेकिन बहुत ही Powerful तरीका है। OOP में एक Computer Program को Objects के एक ऐसे Group के रूप में परिभाषित किया जाता है जिसमें सभी Objects आपस में Interact कर सकते हैं।

यानी सभी Objects अपनी सूचनाओं का एक दूसरे के बीच Transaction कर सकते हैं। OOPS में दुनिया की हर चीज को Object माना जाता है। मानलो कि एक Worker Object Money Object को CompanyFunds Object से लेता है और अपने स्वयं के BankAccount Object में जमा करवाता है। यदि दूसरा कोई Worker Object DoublecheckFund Object को Use करता है तो Polish Object को बुलाया जाता है।

Java Program की सबसे बडी यदि कोई विशेषता है, तो वह ये है कि Java के Program को World Wide Web Pages पर Execute किया जा सकता है। इन Programs को **Applets** कहते हैं। Java से पहले HTML Format में ही विभिन्न Web Pages को लिखा जाता था। ये Web Pages ऐसे होते थे, जिसमें जिस Page को देखना हो उसके Hyperlink पर Click करो और दूसरा Page देखना है तो उसके Hyperlink पर Click करो और आगे से आगे बढते जाओ।

जबकि Java Applet जो कि Web Pages पर Run होते हैं, अधिक अच्छा अनुभव प्रदान करता है। इसमें User के Response के अनुसार Web Page Dynamically Update होता है। जैसे आज हम कई TV Channel पर देखते हैं जहां कोई सवाल पूछा जाता है और लोग SMS भेज कर अपना

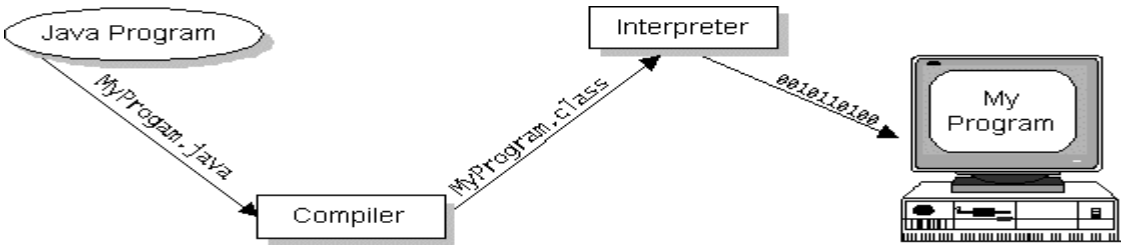
पक्ष बताते हैं। जैसे–जैसे लोग SMS भेजते रहते हैं, SMS की संख्या भी बदलती रहती है। ये काम Dynamically होता है जो कि Java के कारण ही सम्भव हुआ है।

आज User Internet पर उपलब्ध विभिन्न प्रकार के Web Pages से Java के कारण ही Interact कर सकता है। यदि इसका उदाहरण लेना चाहें, तो Share Market का सारा काम Online होता है। User जब चाहे अपने Account की Information को प्राप्त कर सकता है। यदि वह किसी Company का Share खरीदना चाहता है, तो वह Online खरीद सकता है। जैसे ही वह Share खरीदता है, उस Company के Buyers की संख्या बढ जाती है। इसी तरह से यदि Share को बेचा जाता है, तो Company के Sellers की संख्या बढ जाती है। ये जो परिवर्तन Web Pages के Data में होता है, वह Dynamic परिवर्तन कहलाता है। यानी Web Page Dynamically या Run Time में User के Interaction से Update होता है। इस प्रकार की **Secure** Dynamic व Online सुविधा हमें Java के कारण ही प्राप्त हो पा रही है।

हालांकि Web Based Programs की वजह से Java अधिक महत्वपूर्ण लगती है लेकिन ये एक General Purpose Language भी है जिसका प्रयोग सभी तरह के Programs को Develop करने में होता है। आज हम Mobile के जितने भी Software देखते हैं उनमें से ज्यादातर Java Based हैं। Mobile में जो Games Run होते हैं वे ज्यादातर Java में Develop किए जाते हैं।

Java – Working

जब Java के किसी Program को Compile किया जाता है तब Java का Program पूरी तरह से Machine Language में Convert नहीं होता है बल्कि एक Intermediate Language में Convert होता है, जिसे Java Bytecodes कहा जाता है। ये Codes Platform Independent होते हैं, इसलिए इन्हें किसी भी Operating System व किसी भी Processor पर चलाया जा सकता है। Java के Program की Compilation केवल एक ही बार होती है लेकिन जितनी बार भी Java के Program को चलाया जाता है, हर बार उस Program का Interpretation होता है। इसे हम निम्न चित्र द्वारा समझ सकते हैं–

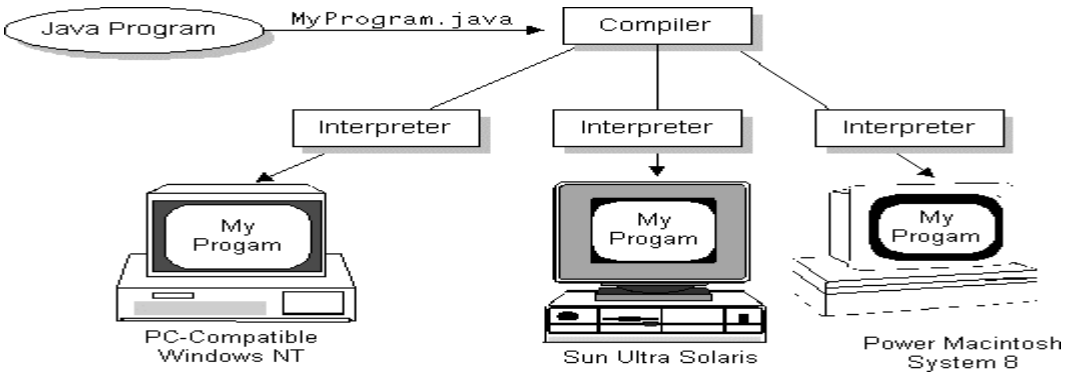


Java Bytecodes को हम Java Virtual Machine (Java VM) के लिए Machine Codes मान सकते हैं। हर Java Interpreter चाहे वह Java Development Tool हो या कोई Browser जो कि Java Applets को Run करता हो, Java Virtual Machine का ही Implementation है।

Java Virtual Machine को Hardware में भी Implement किया जा सकता है, जिसका परिणाम आज के Mobile System Software हैं।

Java Bytecodes हमें ये सुविधा देते हैं कि हम Java के Program को एक बार Compile करें और कहीं भी Run करें। हम किसी Java Program को किसी भी उस Computer पर Compile कर सकते हैं जिस पर Java Compiler हो। फिर उस Java Program के Bytecodes को किसी भी उस Computer पर Run किया जा सकता है जिस पर Java VM हो।

उदाहरण के लिए एक ही Java Program Windows, OS/2 MacOS NT, Macintosh आदि विभिन्न Platform पर Execute हो सकते हैं।

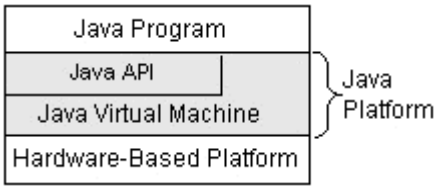


Java Platform

Platform एक एक ऐसा Software या Hardware Environment होता है जिसमें कोई Program Run होता है। Java Platform कई अन्य Platforms से अलग है। Java Platform एक Software Platform है, जो सभी अन्य Hardware Based Platform के Top पर यानी ऊपर Run होता है। ज्यादातर अन्य Platforms Hardware व Operating System का Combination होते हैं।

Java Platform के दो Components हैं। पहला है **Java Virtual Machine (Java VM)** जिसके बारे में हम जान चुके हैं। ये Java Platform का Base या आधार है और विभिन्न Hardware Base Platform के ऊपर रहता है। दूसरा होता है Java Application Programming Interface (Java API) जिसके बारे में हम अब जानेंगे।

Java API Ready Made Software Components का एक बहुत बडा Collection है जो कि Programmer को GUI (Graphical User Interface) जैसी कई उपयोगी क्षमताएं प्रदान करता है। Java API को Related Components की Libraries के रूप में Group कर दिया गया है। इन विभिन्न Related Components के Group को ही **Packages** कहते हैं। एक Java Program को हम निम्न चित्रानुसार दर्शा सकते हैं:



जब एक Java Program को किसी Computer पर Execute किया जाता है तो Java Program व Hardware Based Platform के बीच Java API व Java Virtual Machine की Layer होती हैं जो Java के Program को Hardware Dependencies से अलग करती हैं।

यानी इन दोनों की वजह से Java का कोई Program किसी भी Computer के Hardware पर निर्भर नहीं होता है। एक Platform Independent Environment के रूप में Java का Program अन्य Native Codes Programs की तुलना में कुछ धीमा होता है। लेकिन फिर भी अच्छे Compilers, Java के साथ अच्छी तरह से Tune होने वाले Interpreters और Bytecodes Compilers की वजह से Java की Performance को Native Code की Performance के आस–पास लाया जा सकता है और वो भी जावा की सभी विशेषताओं के साथ।

Java Programs का सबसे अधिक जाना पहचाना यदि कोई रूप है तो वह Java Applets का है। एक Applet भी एक Java Program ही होता है लेकिन इसकी विशेषता ये है कि ये किसी Java Enabled Browser जैसे कि Internet Explorer, Google Chrome, Firefox, Safari, Opera आदि में Run होता है, स्वतंत्र रूप से ये Run नहीं हो सकता। जबकि Java Application Standalone Run हो सकते हैं।

Applets Application के समान ही होते हैं। ऐसा भी नहीं है कि Java का प्रयोग केवल Web Pages Applications लिखने के लिए ही किया जा सकता है। बल्कि Java एक Powerful Software Platform और General Purpose High Level Programming Language भी है।

Java के सबसे Common Application Programs के उदाहरण Servers हैं जो किसी Network के विभिन्न Clients को Service प्रदान करने का काम करते हैं। Web Servers, Proxy Servers, Mail Servers, Print Servers व Boot Servers Java Applications के विभिन्न उदाहरण हैं।

Servlets Applets के समान ही होते हैं, लेकिन किसी Browser में Run होने के बजाय ये Java Servers में Run होते हैं और Java Server की Configuring या Tailoring करते हैं।

एक सवाल पैदा हो सकता है कि Java API इन सभी प्रकार के Programs को किस प्रकार से Support करता है। इसका जवाब ये है कि ये इन सभी प्रकार के Programs को एक Software Components के Package के माध्यम से Support करता है जिसमें विभिन्न प्रकार की

Functionalities होती हैं। Core API एक ऐसा API है जो हर Java Platform में पूरी तरह से Implemented होता है। Core API हमें निम्न Features प्रदान करता है–

The Essentials:

Objects, strings, threads, numbers, input and output, data structures, system properties, date and time जैसी कई चीजों को Handle करने की सुविधा प्रदान करता है।

Applets:

Java applets बनाने के लिए विभिन्न Components प्रदान करता है।

Networking:

Networking की सुविधा प्राप्त करने के लिए URLs, TCP व UDP sockets तथा IP addresses प्रदान करता है।

Internationalization:

ये हमें ऐसी सुविधा प्रदान करता है कि हम ऐसे Programs लिख सकते हैं जो सारी दुनिया में समान रूप से चल सकता है।

Security:

ये हमें Low-level और high-level दोनों प्रकार की सुरक्षा प्रदान करता है। साथ ही electronic signatures, public/private key management, access control और certificates की भी सुविधा प्रदान करता है।

Software components:

ये हमें JavaBeans जैसे Components प्रदान करता है जो किसी पहले से बने हुए Component Architecture में जैसे कि Microsoft's OLE/COM/Active-X architecture, OpenDoc और Netscape's Live Connect में Plug in हो सकता है।

Object serialization:

ये हमें Remote Method Invocation (RMI) द्वारा दूसरे सरल उपकरणों से Communication करने की सुविधा प्रदान करता है, जिसका प्रयोग आज Mobile Technology में भी हो रहा है।

Java Database Connectivity (JDBC):

ये हमें Relational databases से Connect होने व उन्हें Access करने की सुविधा प्रदान करता है।

Java में केवल Core API ही नहीं हैं बल्कि कुछ Standard Extensions भी हैं। ये Standard Extensions 3D, Servers, Collaboration, Telephony, Speech, Animation व कई अन्य चीजों के लिए भी APIs Define करते हैं।

Program

Computer Program एक तरीका है जो Computer को ये बताता है कि उसे कब, क्या करना है। Computer के Boot होने से लेकर Shut Down होने तक जो भी कुछ होता है, किसी ना किसी Program की वजह से होता है। MS-Word एक Program है, Norton Antivirus एक Program है, DOS Prompt पर लिखा जाने वाला हर Command एक Program है, यहां तक कि विभिन्न प्रकार के Computer Viruses भी एक Program हैं।

आज Artificial Intelligence का एक उदाहरण Robots हैं। इन Robots को अमीर लोग अपने घरों में रखते हैं। ये Robots ऐसे होते हैं कि इन्हें जो काम करने के लिए कह दिया जाता है या किसी तरीके से बता दिया जाता है, ये Robots वे सभी काम बडी ही अच्छी तरह से कर लेते हैं।

जैसे यदि आप इन Robots को कहें कि जब आपके घर की Bell Ring हो तो इन्हें घर का दरवाजा खोलना है। तो ये वैसा ही करते हैं। ये Computer Program का एक साधारण सा उदाहरण है जिसमें आप किसी निर्जीव Robot को कुछ Instruction देते हैं, और वह निर्जीव Robot आपकी बात मानता है और आपके द्वारा बताया गया काम कर देता है।

इसी तरह से Computer को भी विभिन्न प्रकार के Instructions प्रदान किए जाते हैं, जिनके अनुसार Computer काम करता है। जैसे कि Microsoft Company ने Windows के Program द्वारा Computer को ये Instruction दिया है कि यदि कोई Mouse को Move करता है, तो Monitor की Screen पर स्थित Cursor या Pointer भी उसी तरह से Move होना चाहिए। यदि कोई Start Button पर Click करता है तो Start Menu Popup होना चाहिए, आदि–आदि।

यानी Computer पर हम जो कोई Action करते हैं, उसे Response करने के लिए पहले से ही Program लिखा गया है। जब कोई **Event** होता है, Computer उस Event से सम्बंधित Program के अनुसार काम करने लगता है और हमें हमारा Required Result प्रदान करता है।

Computer में जो भी कुछ होता है उसे Event कहते हैं। जैसे यदि हम Mouse को Move करते हैं तो MouseMove Event Generate होता है, यदि हम Mouse से Click करते हैं तो

MouseClick, Event Generate होता है। इसी तरह से यदि हम Keyboard पर कोई Key Press करते हैं तो Keypress Event Generate होता है।

ये तो **Hardware Events** के उदाहरण हैं। Computer में Software Events भी Generate होते हैं जिन्हें Response करने के लिए भी विभिन्न प्रकार के Programs लिखे गए हैं। उदाहरण के लिए किसी Window को Minimize करना, Restore करना, किसी Window को Close करना आदि **Software Events** के उदाहरण हैं। निम्न Program देखिए–

```
#include <stdio.h>

main()
{
    printf("Hello Gopala");
}
```

इस Program द्वारा हम हमारे Computer को केवल एक Message Screen पर Print करने के लिए एक Instruction प्रदान कर रहे हैं। ये Program Computer Screen पर “Hello Gopala” Print करता है।

हम किसी Computer Program में जितनी भी Coding Lines लिखते हैं, ये सभी Lines **Program Statements** कहलाती हैं। Computer उन सभी Statements को एक निश्चित क्रम में Handle करता है, ठीक उसी तरह से जिस तरह से एक रसोईया किसी विशेष प्रकार के पकवान को बनाने के लिए एक विशेष क्रम का पालन करता है।

चूंकि Computer उसी क्रम में विभिन्न Statements के अनुसार काम करता है जिस क्रम में एक Programmer किसी Program को लिखता है। इसलिए यदि कोई Program वैसा Result प्रदान नहीं करता, जैसा एक Programmer चाहता है, तो ये Computer की गलती नहीं है बल्कि उस Program की Mistake है।

ज्यादातर Program उसी तरह से लिखे जाते हैं, जिस तरह से हम कोई Letter लिखते हैं, जिसमें किसी Word Processor में हम हर Word को Type करते हैं। कुछ Programming Languages के Compilers के साथ उनके खुद के Word Processors आते हैं, जैसे कि Turbo C++ का Program Creation का पूरा IDE आता है जबकि कुछ Compilers के साथ कोई Word Processor नहीं आता।

जिन Compilers के साथ कोई Word Processor नहीं आता जिसमें Program की Coding की जा सके, तो ऐसे Program के Source Code लिखने के लिए किसी भी अन्य Word Processor

का प्रयोग किया जा सकता है। हम Java Developer Kit के सभी Components का प्रयोग किसी भी Word Processor जैसे कि Notepad या WordPad के साथ कर सकते हैं।

जब एक Program के Source Codes लिख लिए जाते हैं, तो उसके बाद उस Source File को उस Language के Extension के साथ Save करना होता है। जैसे यदि हम Notepad का प्रयोग करके "C" Language का Program लिखते हैं तो File को Save करते समय हमें File के नाम के बाद .C Extension देना होता है। उसी तरह से यदि हम Java के Program को Save करते हैं, तो हमें File के नाम के बाद **.java** Extension का प्रयोग करना होता है। जैसे *Program.java, Application.java* आदि।

हम जो Program लिखते हैं वे English के कुछ सामान्य Words होते हैं। लेकिन Computer केवल Binary Language को ही समझता है। इसलिए हमें एक ऐसे Program की जरूरत होती है जो हमारे Source Codes को Computer के समझने योग्य Machine Language में Convert कर सके।

Interpreter एक ऐसा Program है जो किसी भी Program की Source File के हर Statement या Code की हर Line को Computer की Machine Language में Convert करके Computer को बताता है कि उसे क्या करना है।

कुछ Languages में एक अन्य Software जिसे **Compiler** कहते हैं का प्रयोग करके Source Code File को Machine Language में Convert करता है। इन दोनों में अन्तर केवल इतना है कि Interpreter Source File के हर Line या हर Statement को Computer के समझने योग्य Binary Language में Convert करता है और यदि किसी Statement में कोई Error हो तो उस Line या Statement से आगे Interpret नहीं होता।

जबकि Compiler एक ऐसा Program होता है जो पूरे Program को एक साथ Machine Language में Convert करता है। यदि Program में कोई Error हो तो Program सभी Errors को एक साथ Display करता है और तब तक Program को Machine Language में Convert नहीं करता है जब तक कि सभी Errors को Debug ना कर दिया जाए।

जो Program Interpreted होते हैं वे Compiled Program की तुलना में धीरे चलते हैं। लेकिन Java एक ऐसी Language है जिसको Interpreter व Compiler दोनों की जरूरत होती है।

जब भी हम कोई Program लिखते हैं तो उसमें किसी ना किसी तरह की Errors हमेंशा आती है। इन Errors को Computer Programming की भाषा में **Bug** कहा जाता है और इन Errors को सही करने के Process को **Debug** करना कहते हैं।

Procedural Techniques and OOPS

Pascal, C, Basic, Fortran जैसी पारम्परिक भाषाएं Procedural Languages के उदाहरण हैं, जिसमें प्रत्येक Statement Computer को कुछ करने का आदेश देता है। यानी Procedural Languages Instructions का एक समूह होता है। Procedural Languages में छोटे Programs के लिये किसी भी अन्य प्रकार के Pattern की आवश्यकता नही होती है। Programmer Instructions की List बनाता है और Computer उनके अनुसार काम करता है।

जब प्रोग्राम काफी बडे व जटिल हो जाते हैं, तब Instructions की यह List काफी परेशानी पैदा करती है। इसलिये एक बडे प्रोग्राम को छोटे–छोटे टुकडों में बांट दिया जाता है। इन छोटे–छोटे टुकडों को Functions कहा जाता है। Functions को दूसरी अन्य भाषाओं में Subroutine, Subprogram या Procedure कहा जाता है।

एक बडे प्रोग्राम को छोटे–छोटे Functions में विभाजित करने से पूरा Program Functions का एक समूह बन जाता है। इसे Module कहा जाता है। लेकिन ये Modules भी Procedural Programming में ही आते हैं क्योंकि सभी Functions में Statements की एक List होती है और सभी Functions मिल कर पूरा Program बनाते हैं, जिससे पूरा Program Instructions की एक बहुत बडी List बन जाती है।

Procedural Languages के शुरूआती दौर में इनमें ही Program Develop किए जाते थे। "C" भी एक Procedural Languages है और जब "C" भाषा का आविष्कार हुआ था तब Programmers अन्य भाषाओं को छोड कर "C" में ही अपने Program Develop करने लगे थे।

लेकिन समय व आवश्यकता के अनुसार जब Program बडे व जटिल होने लगे, तब Programmers को इस भाषा में प्रोग्राम बनाने में दिक्कतें आने लगीं। उन्होने महसूस किया कि इस भाषा में कुछ सुधार की आवश्यकता है ताकि ये भाषा सरल व लोकप्रिय बन सके। ये भाषा सरल बन सके इसके लिये इसका वास्तविक जीवन के अनुसार होना जरूरी था।

यानी हम हमारे सामान्य जीवन में जिस प्रकार से व्यवहार करते हैं, इस भाषा का भी वैसा ही होना जरूरी था ताकि Programmers इसमें अधिक सरलता व सफलता से Program बना सकें। भाषा वास्तविक जीवन के अनुसार हो, यही **Concept Object Oriented Programming** यानी **OOP** का आधार बना। "C" भाषा की इन कमियों को पहचाना गया और इसमें सुधार किया गया।

फलस्वरूप हमें "C" भाषा का एक नया संस्करण "C++" प्राप्त हुआ जो कि Object Oriented Concept पर आधारित है। आवश्यकता के अनुसार इस भाषा की कमियों को भी पहचाना गया और उसमें सुधार करने पर जो नई भाषा सामने आई वह Java थी। आइयें, हम भी जानने की कोशिश करते हैं कि "C" भाषा में ऐसी कौनसी कमियां थीं, जिनमें सुधार की आवश्यकता महसूस की गई ?

Procedural Languages में काम होने का महत्व था Data का नहीं, यानी कि Keyboard से Data Input किया जाए, Data पर Processing की जाए, Errors को Check किया जाए आदि। Functions में भी इसी महत्व को जारी रखा गया। Functions कोई काम करते हैं, उसी प्रकार से जिस प्रकार से साधारण Statement करता है। Functions कोई जटिल काम भी कर सकते हैं लेकिन इनमें भी काम के होने का ही महत्व था।

पूरे Program में Data पर कोई ध्यान नहीं दिया जाता था जबकि पूरे प्रोग्राम का मूल आधार Data ही होता है। किसी Inventory के Program में इस बात का कोई ज्यादा महत्व नहीं होता है कि Data को किस प्रकार से Display किया जाता है या एक Function किस प्रकार से Corrupt Data को Check करता है, बल्कि इस बात का होता है कि Data क्या है और वह किस प्रकार से Program में काम आ रहा है। Procedural Program में Data को द्वितीय स्तर पर रखा गया था जबकि किसी भी Program का मूल आधार Data ही होता है।

उदाहरण के लिये, किसी Inventory के Program में किसी Data File को Memory में Load किया जाता है, तब ये File एक Global Variable की तरह होती है, जिसे कोई भी Function Use कर सकता है। ये Functions Data पर विभिन्न प्रकार के Operations करते हैं। यानी ये Data को Read करते है, Analyze करते हैं, Update करते हैं, Rearrange करते हैं, Display करते हैं और वापस Disk पर Write करते हैं। "C" में Local Variables भी होते हैं लेकिन Local Variables, महत्वपूर्ण Data के लिये इतने उपयोगी नहीं होते हैं, जो कि विभिन्न Functions द्वारा Access किए जाते हैं।

मान लों कि एक नए Programmer को Data को किसी खास तरीके से Analyze करने के लिये एक Function लिखने को कहा गया। प्रोग्राम की जटिलता से अनभिज्ञ Programmer एक ऐसा Function बनाता है, जो कि अचानक किसी महत्वपूर्ण Data को नष्ट कर देता है। ऐसा होना काफी आसान है क्योंकि कोई भी Function Data को Access कर सकता है।

इसलिये क्योंकि Procedural Language में Data Global होता है। ये कुछ ऐसा ही है जैसे कि आप अपने Personal कागजात को Telephone Directory के पास रख दें जहां कभी भी कोई भी पहुंच सकता है, उससे छेडछाड कर सकता है और उसे नष्ट कर सकता है। इसी प्रकार से Procedural Languages में होता है जहां आपका Data Global होता है और कोई भी Function उसे Use करके खराब कर सकता है या नुकसान पहुंचा सकता है।

Procedural Languages की दूसरी कमी ये थी कि कई Functions एक साथ एक ही Data को Use कर रहे होते हैं, इसलिये Data को Store करने का तरीका काफी जटिल हो जाता है। समान Data को Use कर रहे सभी Functions को Modify किए बिना Data में किसी प्रकार का कोई परिवर्तन नहीं किया जा सकता है।

उदाहरण के लिये यदि आप एक नया Data Add करते हैं तो उन सभी Functions को Modify करना होगा जो कि Data को Use कर रहे हैं, ताकि ये सभी Functions Add किए गए नए Data

को Use कर सकें। ये पता करना कि कौन–कौन से Function Data को Use कर रहे हैं और सभी को बिल्कुल सही तरीके से Modify करना काफी कठिन होता है।

Procedural Programs को Design करना काफी मुश्किल होता है। समस्या ये होती है कि इनका Design वास्तविक जीवन से Related नहीं होता है। जैसे कि, माना आप एक Graphics User Interface में Menus, Windows के लिये Code लिखना चाहते हैं, तो आपको ये तय करना मुश्किल होगा कि कौनसे Functions Use किए जाए? कौनसा Data Structure Use किया जाए? आदि। इनका कोई स्पष्ट उत्तर नहीं है।

Procedural Programs के साथ कई और परेशानियां हैं। उनमें से एक समस्या नए Data Type की है। Computer Languages में कई प्रकार के Built-in Data Types होते हैं, जैसे कि Integer, Float, Character आदि। मानलो कि आप Complex Numbers के साथ प्रक्रिया करना चाहते हैं या Two-dimensional Coordinates के साथ काम करना चाहते हैं या Data के साथ प्रक्रिया करना चाहते हैं। Built-in Data Type इनको आसानी से Handle नहीं कर सकते हैं।

इसलिए हमें हमारी आवश्यकतानुसार स्वयं के Data Type बनाने की जरूरत होती है। Procedural Language में स्वयं के Data Type बना कर हम उन्हे बिल्कुल Built-in Data Type की तरह Use नहीं कर सकते हैं। Procedural Language इतने उन्नत नहीं हैं। बिना अप्राकृतिक जटिल तरीकों के आप Procedural Languages में x व y दोनों Coordinates को एक ही Variable में Store करके उस पर Processing नहीं कर सकते हैं। Procedural Languages को लिखना व Maintain करना काफी मुश्किल काम होता है।

The Object-Oriented Approach

Object Oriented Language का मूलभूत विचार ये है कि जिस समस्या का समाधान Computer पर प्राप्त करना है उस समस्या के मूल Data और उस Data पर काम करने वाले Functions को Combine करके एक Unit के रूप में ले लिया जाता है। इस Unit को **Object** कहा जाता है।

एक Object के Data पर काम करने के लिये लिखे गए Operations या Functions को Java में **Methods** कहा जाता है। ये Methods किसी Object के Data को Access करने का एक मात्र माध्यम होते हैं। यदि आप किसी Object के अन्दर रखे किसी Data को Read करना चाहते हैं, तो आपको इसी Object के अन्दर Define किए गए उस Method को Use करना पडता है, जिसे उस Object के Data को Access करने के लिये ही परिभाषित किया गया है। यही एक Method होता है जिसकी मदद से आप उस Object के Data को Read कर सकते हैं।

आप सीधे ही Data के साथ किसी प्रकार की प्रक्रिया नहीं कर सकते हैं क्योंकि Data Hidden रहता है। इसलिये किसी प्रकार से अचानक हुए परिवर्तन से Data सुरक्षित रहता है। Data व Data को

Use कर सकने वाले Functions या Operations का एक साथ एक ही Unit के रूप में होना **Encapsulation** कहलाता है।

Data का Hidden रहना यानी **Data Hiding** व **Encapsulation** Object Oriented Programming का मूल तथ्य या Key Terms है। यदि आप किसी Data को Modify करना चाहते हैं, तो आपको पता होना चाहिए कि कौनसा Method उस Data पर Required Operation करने की क्षमता रखता है। कोई भी अन्य Method उस Data को Access नहीं कर सकता है। ये Processing Program को लिखना, Debug करना व Maintain करना आसान बनाती है।

एक Java का प्रोग्राम ढेर सारे विभिन्न प्रकार के Objects का बना होता है, जो कि अपने–अपने Methods द्वारा आपस में Communication करते हैं। Java व कई अन्य OOP Languages में Member Functions को **Methods** और Data Item को **Instance Variable** कहा जाता है। किसी Object के Methods को Use करना उस Object को **Message Send** करना कहलाता है।

हम एक उदाहरण लेते हैं। माना एक बडा प्रीति–भोज का समारोह है जिसमें सभी मेहमान किसी Dining Table के चारों ओर बैठे हैं। जो भी खाना Table पर रखा है हम उसे Data कह सकते हैं और उस Table के चारों ओर बैठे लोगों को हम Functions या Operations मान सकते हैं।

Object के Operations हमेंशा अपने Data यानी Attributes की State में परिवर्तन करते हैं। इस उदाहरण में Data (खाना) पर खाना खाने का Operation Perform किया जा रहा है। इस व्यवस्था में जब भी किसी को Table पर रखे विभिन्न प्रकार के व्यंजनों में से कुछ लेना होता है, तो वह स्वयं ही उस व्यंजन तक पहुंचता है और उसे उपयोग में ले लेता है। किसी पडौसी मेहमान से कोई भी व्यंजन Pass करने को नहीं कहता।

Procedural Program का भी यही तरीका होता है। ये तरीका तब तक बहुत ठीक है, जब तक कि खाना खाने वाले मेहमानों की संख्या सीमित हो। लेकिन यदि मेहमानों की संख्या अधिक हो तो ये तरीका ठीक नहीं कहा जा सकता है।

क्योंकि जब मेहमान अधिक होंगे तो Table भी बडा होगा और खाने के विभिन्न सामान पूरे Table पर काफी दूर–दूर होंगे। ऐसे में यदि कोई मेहमान किसी दूर रखे व्यंजन तक पहुंचने की कोशिश करता है, तो हो सकता है कि कोशिश करते समय उसके Shirt की Sleeves किसी दूसरे मेहमान के खाने में चली जाए या ऐसा भी हो सकता है कि कई मेहमान एक साथ किसी एक ही व्यंजन पर हाथ बढाएं और व्यंजन Table पर गिर कर खराब हो जाए।

यानी यदि मेहमानों की संख्या काफी ज्यादा हो तो एक ही Table पर भोजन करना एक परेशानी वाला काम होगा। एक बडे **Procedural Program** में भी यही होता है।

इस समस्या के समाधान के रूप में यदि कई छोटे–छोटे Tables हों और उन पर एक सीमित मात्रा में मेहमान हों और सबके पास उनका अपना भोजन हो, तो ये एक अच्छी व्यवस्था हो सकती है। इस छोटे Table पर सभी मेहमान किसी भी व्यंजन पर आसानी से पहुंच सकते हैं। यदि कोई मेहमान किसी अन्य Table पर रखे किसी व्यंजन को लेना चाहता है तो सम्भवतया वह किसी अन्य मेहमान से उस व्यंजन को लाने के लिये कह सकता है।

ये तरीका **Object Oriented Programming** का है जिसमें हरेक छोटी Table को एक **Object** कहा जा सकता है। हरेक Object में उसका स्वयं का **Data** और Data पर Perform होने वाला **Operation** या **Function** होता है। Data व Operations के बीच होने वाले विभिन्न लेन–देन अधिकतर Object के अन्दर ही होते हैं लेकिन आवश्यकतानुसार ये भी सम्भव है कि किसी अन्य Object के Data को भी Use किया जा सके।

चूंकि एक Object के Data को केवल वही Object Access कर सकता है, इसलिए यदि किसी Object A के Data को कोई दूसरा Object B Access करना चाहता है, तो वह Object B Object A से Data को Access करने के लिए कहता है। इस प्रक्रिया को **Message Passing** करना कहते हैं। Object A Object B की Request को पूरा करता है और अपने Data को Access करके करने के लिए उस दूसरे Object B को दे देता है।

Difference Between C++ and Java

वास्तव में Java "C" व "C++" का ही Modified रूप है। चूंकि आज भी ज्यादातर Professional लोग बडे Projects के लिए "C++" को ही चुनते हैं, इसलिए ये जानना जरूरी है कि Java में "C++" की किन विशेषताओं को लिया गया है और किन चीजों को छोडा गया है जो सामान्य Programmer को परेशान करती हैं।

Preprocessor

"C" व "C++" में Program के Compilation को Control करने के लिए Preprocessors का प्रयोग किया जाता है। "C++" का Compiler किसी भी Source Program को Compile करने से पहले सभी Preprocessor Directives को Expand करने का काम करता है। सभी "C" व "C++" के Programmers जानते हैं कि Preprocessors का प्रयोग करने पर Program की जटिलता बढ जाती है। "C++" के Programmer Preprocessors का प्रयोग करके लगभग स्वयं की Language बनाना शुरू कर देते हैं।

ज्यादातर Statement के लिए व Constant मानों के लिए वे Preprocessors का प्रयोग करते हैं। इससे Program की जटिलता इतनी बढ जाती है कि कोई भी नया Programmer यदि उस Program को समझना चाहे तो उसे काफी परेशानी आती है। साथ ही इन Program Codes को Reuse भी नहीं किया जा सकता है।

Preprocessor Directives की एक कमी ये भी है कि इनकी Type Checking कभी भी निश्चित नहीं होती। यानी ये हमेंशा एक String Format को Follow करते हैं। यदि हम ***#define MAX 10*** Statement लिखते हैं, तो यहां मान 10 Integer नहीं बल्कि एक String होता है।

Java में Preprocessors को हटा दिया गया है। हालांकि Java Preprocessor Directives के समान ही Functionality प्रदान करता है लेकिन अधिक Control के साथ। Java में #define के स्थान पर Constant Data Members का प्रयोग किया जाता है।

इसका परिणाम ये है कि Java के Codes को पढना व समझना "C++" के Codes को पढने व समझने की तुलना में अधिक सरल हो जाता है। साथ ही Java के Programs में Header Files का प्रयोग नहीं होता है बल्कि Java का Compiler Source Code File से सीधे ही Class Definitions बना लेता है जिसमें Class Definitions व Methods दोनों होते हैं।

Pointers

जितने भी "C" या "C++" के Programmers हैं, वे सभी मानते हैं कि यदि Pointers को पूरी सावधानी से प्रयोग ना किया जाए तो ऐसे Errors Generate होते हैं, जिन्हें Debug करने में दिमाग का पसीना निकल जाता है। साथ ही Pointers के प्रयोग से Program हमेंशा समझने में जटिल हो जाता है, हालांकि Pointers के प्रयोग से हमारा Program Directly Memory Locations को Access कर सकता है, इसलिए Program की Speed तुलना में तेज हो जाती है।

"C++" के Programmers हमेंशा Dynamic Data Structure को Create व Maintain करने के लिए Pointers Arithmetic का प्रयोग करते हैं और हमेंशा जटिल **Bugs** में फंसते हैं। "C++" Programmers का ज्यादातर समय उन Programs को Create करने में नहीं बीतता जिनमें Pointer का प्रयोग होता है, बल्कि उन Programs को Debug करने में बीतता है।

Java Pointers को Support नहीं करता है। यानी Java में Pointes जैसी कोई व्यवस्था नहीं है जो Directly Memory को Access कर सके। हालांकि Pointers के स्थान पर Java में **References** का बहुत प्रयोग किया जाता है जो कि Pointers के समान ही काम करते हैं लेकिन References का Arithmetic सामान्य Arithmetic जैसा ही होता है ना कि Pointer Arithmetic जैसा।

इस Process से उन सभी Errors से छुटकारा मिल जाता है जो Pointers के Mismanagement के कारण Generate होती हैं। References का प्रयोग करने से Java के Program पर्याप्त Readable व समझने योग्य होते हैं जबकि Pointers का चाहे पूरी तरह से सही प्रयोग किया जाए, लेकिन Program आसानी से समझने योग्य व Readable नहीं होता है।

"C" व "C++" के Programmers सोंच सकते हैं कि वे जो काम Pointers का प्रयोग करके काफी आसानी से कुछ Data Structures को Implement कर सकते थे वे काम Java में नहीं किए जा सकेंगे। जैसे कि Dynamic Arrays Java में Create नहीं हो सकते। लेकिन ऐसा नहीं है। वास्तविकता ये है कि वे सभी काम Java में Objects व Objects के Array के प्रयोग से अधिक आसानी व Reliability के साथ किए जा सकते हैं।

Java हमें कुछ Runtime Security भी प्रदान करता है जो कोई अन्य Language Provide नहीं करती। जैसे कि यदि हम "C" या "C++" में किसी Array की Size को 10 Define किया है और हम 11^{th} Index Number पर कोई मान Input करना चाहें तो Java हमें ऐसा नहीं करने देता जबकि "C" व "C++" में हम ऐसा करके किसी दूसरे Data को Damage कर सकते हैं।

Structure and Union

"C" में दो तरह के (Structure and Union) और "C++" में तीन तरह के (Structure Union and Class) Complex Data Types हैं। Java में केवल एक ही Complex Data Type है जिसे Class कहते हैं। "C" व "C++" में जितने काम इन तीनों को प्रयोग करके किए जाते हैं Java में वे सभी काम केवल एक Class से ही किया जा सकता है। जब हमें Structure या Union के Functionality की जरूरत होती है तो Java हमें इन Functionality को Class द्वारा प्राप्त करने के लिए बाध्य करता है।

हालांकि ऐसा लग सकता है कि Java में Programmers को Structure व Union के स्थान पर Class Use करने के लिए Extra काम करना पडेगा लेकिन ऐसा नहीं है। बल्कि Class के प्रयोग से Program की जटिलता कम हो जाती है।

Java बनाने वालों ने इसी बात को प्राथमिकता दी है कि जितना हो सके उतना Java को Simple व आसानी से समझने योग्य Language में रखा जाए, इसलिए Java में से "C" व "C++" की उन चीजों को हटा दिया गया है जो सीखने व समझने में परेशानी Create करती हैं और Language को जटिल बनाती हैं।

वैसे "C" व "C++" को उन लोगों के लिए लिखा गया था जो पहले से ही पेशेवर Programmer थे, लेकिन Java को नए Programmers को ध्यान में रख कर Develop किया गया है, जो कि पेशेवर नहीं हैं और Computer को बहुत गहराई से नहीं जानते बल्कि जिन्हें Computer का सामान्य ज्ञान ही है।

चूंकि Java में Structure व Union नहीं हैं इसलिए Java एक पूर्ण Object Oriented Programming Language है, क्योंकि इसमें जो भी काम करना होता है, उसके लिए Class Develop करनी पडती है। इससे Program Codes को ना केवल Reuse किया जा सकता है बल्कि Program को Maintain करना भी अन्य Languages की तुलना में काफी सरल होता है।

Functions

"C" व "C++" में सभी Program Codes को किसी ना किसी Function में लिखा जाता है। यहां तक कि Main Program भी एक Function main() में होता है जहां आवश्यकतानुसार अन्य Functions को Call करके अपना काम पूरा किया जाता है। ये Functions Global रखे जाते हैं ताकि पूरे Program का कोई भी अन्य Function इन्हें Call कर सके और Data पर Required Processing कर सके। "C++" में भी Functions होते हैं जिनके माध्यम से Class के Data को Access किया जाता है। इन Functions को ही Method कहा जाता है।

"C++" Class के Methods Java Class के Methods के समान ही होते हैं। फिर भी चूंकि "C++" "C" जैसी Procedural Language को भी Support करता है इसलिए इसे पूरी तरह से Object Oriented Programming Language नहीं कहा जा सकता, बल्कि इसे Hybrid Language कहा जाता है। इस स्थिति में "C++" के Class के Methods व Class के बाहर के Functions दोनों का Mixture होता है, इसलिए कुछ हद तक "C++" में Confusion की स्थिति भी बनी रहती है जिससे कई बार अजीब तरह के Bugs का सामना करना पडता है।

Java में कोई Function नहीं होता है, इसलिए Java "C++" की तुलना में पूरी तरह से Object Oriented Programming Language है। Java Programmer को इस बात के लिए बाध्य करती है कि Programmer अपने सभी Routines को Class Methods के रूप में व्यवस्थित करके रखे। Java की इस बाध्यता के कारण Programmer अपने Codes को अधिक अच्छे तरीके से Organize करके रखता है।

Multiple Inheritance

Multiple Inheritance "C++" की एक ऐसी व्यवस्था है जिसमें हम कई Classes से अपनी आवश्यकतानुसार कुछ–कुछ Features को लेकर एक नई Class बना सकते हैं। यानी हम एक Class को कई Parent Class से Derive कर सकते हैं। हालांकि Multiple Inheritance वास्तव में काफी Powerful है, लेकिन कई Classes से एक Derive Class बनाने व उसे Maintain करने में Programmer को काफी परेशानी का सामना करना पडता है।

Java Multiple Inheritance को Support नहीं करता है। "C++" की इस Functionality को हम Java में Interfaces का प्रयोग करके प्राप्त कर सकते हैं। Java के **Interfaces** Object Method Description Provide करते हैं लेकिन इनमें कोई Implementation नहीं होता है।

Strings

"C" व "C++" में Text Strings को Handle करने के लिए कोई Built – In Support नहीं है। String को Handle करने के लिए "C" व "C++" Programmers को एक Null Terminated Array का प्रयोग करना पडता है।

Java में Strings को एक First Class Object की तरह Implement किया जाता है यानी, Strings Java Language का मुख्य बिन्दु या Core है। Java में हम एक Object के रूप में String को Implement करते हैं, जिससे हमें कई Advantages प्राप्त होती है।

goto Statement

"C" व "C++" में इस Statement को आवश्यकतानुसार काफी Use किया जाता है। लेकिन इसके Use करने पर जितनी तरह की परेशानियां आती हैं, उन्हें तय करना ही मुश्किल है। goto Statement का प्रयोग यदि बडे Program में किया जाता है, तो किसी नए Programmer के लिए उस Program को समझना नामुमकिन हो जाता है। ये एक Branching Statement है, लेकिन Program में ये Statement Program Control को काफी Jump करवाता है, जिससे Program की Efficiency भी प्रभावित होती है।

Java में इस Statement को Support नहीं किया गया है। Java ने goto को एक Keyword के रूप में मान्यता दी है लेकिन इसके उपयोग को Support नहीं किया है।

Operator Overloading

Operator Overloading को Java में Support नहीं किया गया है। Operator Overloading "C++" की एक ऐसी तकनीक है जिससे किसी भी Primary Operator को आवश्यकतानुसार दूसरा अर्थ प्रदान कर दिया जाता है। जैसे + Operator का प्रयोग दो Objects को जोडने के लिए किया जा सकता है। हालांकि Java में इस काम को Class में किया जाता है। लेकिन Operator Overloading तकनीक को Java में छोड दिया गया है। Operator Overloading Programmer के लिए एक सुविधा होती है, लेकिन इससे एक ही Operator को एक ही Program में विभिन्न अर्थ प्रदान कर दिए जाने से Program को समझने में Confusion हो जाने की सम्भावना रहती है। इसलिए इस तकनीक को Java में छोड दिया गया है।

Automatic Type Casting

"C" व "C++" में Automatic Type Casting हो सकती है। यानी एक Integer प्रकार के Variable को यदि Float प्रकार का मान प्रदान करना हो, तो "C" व "C++" का Compiler Automatic Type Casting करके एक Variable के मान को दूसरे Variable के मान में

Automatically Convert कर देता है। यह Automatic Type Casting कहलाती है। जबकि Java में इसे Support नहीं किया गया है।

क्योंकि जब किसी एक Data Type के Variable को दूसरे प्रकार के Data Type के Variable का मान प्रदान किया जाता है, तो Data के मान का Loss हो जाता है। जैसे यदि Integer प्रकार के Variable को किसी Float प्रकार के Variable का मान प्रदान किया जाए, तो Integer प्रकार के Variable में Float प्रकार के मान में दसमलव के बाद की संख्या Store नहीं होती है। यानी दसमलव के बाद की संख्या का Loss हो जाता है।

इसलिए Java में इस Automatic Type Casting को Support नहीं किया गया है। Java में यदि किसी Data के मान का किसी भी प्रकार की प्रक्रिया से Loss होता है तो Java Automatic Type Casting नहीं करता है। इसकी Type Casting Programmer को ही करनी पडती है।

Variable Number of Arguments

"C" व "C++" में हमने देखा है कि हम printf() Function या Scanf() Function में अपनी आवश्यकतानुसार एक या एक से अधिक चाहे जितने भी Arguments प्रदान कर सकते हैं। यदि हमें चार Variable के मान Output में Print करने हों तो हम printf() Function में चार Variable Argument के रूप में भेज सकते हैं और यदि केवल एक ही Variable का मान Print करना हो तो केवल एक Argument भी printf() Function में भेज सकते है। हालांकि ये एक अच्छा तरीका है लेकिन Java में इसे Support नहीं किया गया है। क्योंकि Compiler ये कभी Check नहीं कर पाता है कि उसे जिस Argument के मान को Print करना है वह उचित Data Type का है या नहीं।

Command Line Argument

Java Program में जिस तरह से Command Line पर Argument Pass किए जाते हैं वे "C" व "C++" के Program में Pass किए जाने वाले Arguments की तुलना में थोडे अलग होते हैं। "C" व "C++" में System main() Function को दो Arguments Pass करता है। पहला Argument **argc** और दूसरा Argument **argv** होता है। पहला Argument argc ये Specify करता है कि दूसरे Argument argv में कितने Arguments हैं। दूसरा Argument argv एक Array of Characters का Pointer होता है। इसी Argument में Command Prompt पर दिए जाने वाले Actual Arguments होते हैं।

Java में System Command Line से केवल एक ही Argument args Java Program में Pass करता है। ये एक Strings का Array होता है जिसमें Command Line Arguments होते हैं।

"C" व "C++" में Command Line Arguments एक Program में Pass होते हैं। इस Argument में उस Program का नाम भी होता है जिसमें Arguments को Pass करना होता है। ये नाम Command Line पर सबसे पहले First Argument के रूप में लिखा जाता है। Java में हम पहले से ही उस Program का नाम जानते होते हैं जिसमें हमें Argument भेजना होता है। ये नाम हमेंशा उस Program के Class का नाम ही होता है, इसलिए Command Prompt पर हमें केवल Argument ही Pass करना होता है।

Programming – The Basic Concept

Computer Programming समझने से पहले हमें ये समझना होता है कि Computer क्या काम करता है और कैसे काम करता है। कम्प्युटर का मुख्य काम Data का Management करना होता है। हमारे आस–पास जो भी चीजें हमें दिखाई देती हैं वे सभी एक Object Oriented Programming Language के लिए Objects हैं।

हर Objects के कुछ Attributes होते हैं, जिनसे किसी Particular Object की पहचान होती है। इस Attribute में Numerical या Non–Numerical किसी ना किसी प्रकार का मान प्रदान किया जा सकता है। ये मान ही Computer का वह Data होता है, जिसे Computer Manage करता है।

यदि बिल्कुल ही सरल शब्दों में कहें तो कोई मान या मानों का समूह Computer के लिए Data होता है। ये मान दो तरह के हो सकते हैं : Numerical या Non-Numerical, Computer में हम इन्हीं Data को Store करते हैं, Process करते हैं और किसी ना किसी प्रकार की Information Generate करते हैं।

Computer केवल Electrical Signals या मशीनी भाषा को समझता है। ये मशीनी भाषा बायनरी रूप में होती है जहां किसी Signal के होने को 1 व ना होने को 0 से प्रदर्शित किया जाता है। यदि हम हमारी किसी बात को Binary Format में Computer में Feed कर सकें, तो Computer हमारी बात को समझ सकता है।

Computer भाषा वह भाषा होती है, जिसे Computer समझ सकता है। हर Computer भाषा का एक Software होता है, जिसे Compiler या Interpreter कहते हैं। ये Software हमारी बात को Computer के समझने योग्य मशीनी भाषा या Binary Format में Convert करता है।

Computer को कोई बात समझाने के लिए उसे एक निश्चित क्रम में सूचनाएं देनी होती हैं, जिन्हें Instructions कहा जाता है। Computer इन दी गई Instructions के अनुसार काम करता है और हमारी इच्छानुसार हमें परिणाम प्रदान करता है। Instructions के इस समूह को ही Program कहा जाता है।

Computer में हर Electrical Signal या उसके समूह को Store करके रखने की सुविधा होती है। इन Electrical Signals के समूह को **File** कहते हैं। Computer में जो भी कुछ होता है वह File के रूप में होता है। Computer में दो तरह की File होती है। पहली वह File होती है जिसमें हम हमारे महत्वपूर्ण Data Store करके रखते हैं। इसे **Data File** कहा जाता है।

दूसरी File वह File होती है, जिसमें Computer के लिये वे Instructions होती हैं, जो Computer को बताती हैं कि उसे किसी Data पर किस प्रकार से Processing करके Result Generate करना है। इस दूसरी प्रकार की File को **Program File** कहा जाता है।

हम विभिन्न प्रकार की Computer Languages में Program Files ही Create करते हैं। जब बहुत सारी Program Files मिल कर किसी समस्या का समाधान प्राप्त करवाती हैं, तो उन Program Files के समूह को Software कहा जाता है। Computer Software मुख्यतया दो प्रकार के होते हैं:

System Software:

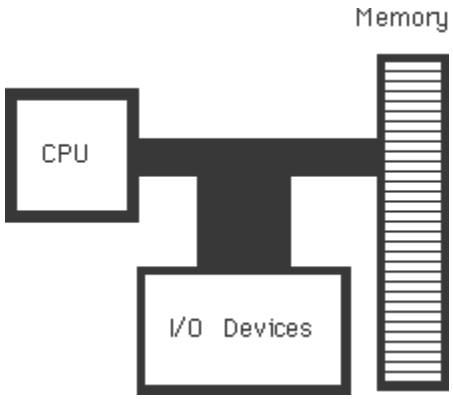
ये Software उन प्रोग्रामों का एक समूह होता हैं जो कम्प्यूटर की Performance को Control करता है। यानी Computer पर किस तरह से एक प्रोग्राम रन होगा और किस तरह से प्रोग्राम Output देगा। किस तरह Hard Disk पर Files Save होंगी, किस तरह पुनः प्राप्त होंगी, आदि। Windows, Unix, Linux, आदि System Software के उदाहरण हैं।

Application Software:

ये Software प्रोग्रामरों द्वारा लिखे जाते हैं व ये Software किसी खास प्रकार की समस्या का समाधान प्राप्त करने के लिये होते हैं। जैसे Tally, MS–Office आदि Application Software के उदाहरण हैं।

Computer Architecture

Computer से अपना मनचाहा काम करवाने के लिए, सबसे पहले हमें Computer के Architecture को समझना होगा। Computer के Architecture को समझे बिना, हम Computer Programming को ठीक से नहीं समझ सकते। Computer System के मुख्य–मुख्य तीन भाग होते हैं–



I/O Devices

वे Devices जिनसे Computer में Data Input किया जाता है और Computer से Data Output में प्राप्त किया जाता है, I/O Devices कहलाती हैं। Keyboard एक Standard Input Device है और Monitor एक Standard Output Device है।

Center Processing unit (CPU)

यह एक Microprocessor Chip होता है। इसे Computer का दिमाग भी कहा जाता है क्योंकि Computer में जो भी काम होता है, उन सभी कामों को या तो CPU करता है या Computer के अन्य Devices से उन कामों को करवाता है। इसका मुख्य काम विभिन्न प्रकार के Programs को Execute करना होता है। इस CPU में भी निम्न विभाग होते हैं जो अलग–अलग काम करते हैं–

Control Unit

इस Unit का मुख्य काम सारे Computer को Control करना होता है। CPU का ये भाग Computer की आंतरिक प्रक्रियाओं का संचालन करता है। यह Input/Output क्रियाओं को Control करता है, साथ ही ALU व Memory के बीच Data के आदान–प्रदान को निर्देशित करता है।

यह Program को Execute करने के लिए Program के Instructions को Memory से प्राप्त करता है और इन Instructions को Electrical Signals में Convert करके उचित Devices तक पहुंचाता है, जिससे Data पर Processing हो सके। Control Unit ALU को बताता है कि Processing के लिए Data Memory में कहां पर स्थित हैं, Data पर क्या प्रक्रिया करनी है और Processing के बाद Data को वापस Memory में कहां पर Store करना है।

Arithmetic Logic Unit (ALU)

CPU के इस भाग में सभी प्रकार की अंकगणितीय व तार्किक प्रक्रियाएं होती हैं। इस भाग में ऐसा Electronic Circuit होता है जो Binary Arithmetic की गणनाएं करता है। ALU Control Unit से निर्देश या मार्गदर्शन लेता है, Memory से Data प्राप्त करता है और परिणाम को या Processed Data को वापस Memory में ही Store करता है।

Registers

Microprocessor में कुछ ऐसी Memory होती है जो थोडे समय के लिए Data को Store कर सकती है। इन्हें Registers कहा जाता है। Control Unit के निर्देशानुसार जो भी Program Instructions व Data Memory से आते हैं वे ALU में Calculation के लिए इन्हीं Registers में Store रहते हैं। ALU में Processing के बाद वापस ये Data Memory में Store हो जाते हैं।

Memory

Memory Computer की Working Storage या कार्यकारी मेमोरी होती है। यह Computer का सबसे महत्वपूर्ण भाग होता है। इसे RAM कहते हैं। इसी में Process होने वाले Data और Data पर Processing करने के लिखे गए Program Instructions होते हैं, जिन्हें Control Unit ALU में Processing के लिए Registers में भेजता है। Processing के बाद जो सूचनाएं या Processed Data Generate होते हैं, वे भी Memory में ही आकर Store होते हैं।

Memory में Data को संग्रह करने के लिए कई Storage Locations होती हैं। हर Storage Location एक Byte की होती है और हर Storage Location का एक पूर्णांक Number होता है, जिसे उस Memory Location का Address कहते हैं। हर Storage Location की पहचान उसके Address से होती है। 1 Byte की RAM में एक ही Character Store हो सकता है और इसमें सिर्फ एक ही Storage Location हो सकती है।

इसी तरह 1 KB की RAM में 1024 Storage Locations हो सकती हैं और इसमें 1024 अक्षर Store हो सकते हैं। जो Memory जितने Byte की होती है उसमें उतने ही Characters Store हो सकते हैं और उसमें उतनी ही Storage Locations हो सकती हैं।

जिस तरह से किसी शहर में ढेर सारे घर होते हैं और हर घर का एक Number होता है। किसी भी घर की पहचान उसके घर के Number से भी हो सकती है। उसी तरह से Memory में भी विभिन्न Storage Cell होते हैं जिनका एक Unique Number होता है। हम किसी भी Storage Cell को उसके Number से पहचान सकते हैं और Access कर सकते हैं। हर Storage Cell के इस Unique Number को उस Storage Cell का Address कहते हैं।

जिस तरह से हम किसी घर में कई तरह के सामान रखते हैं और जरूरत होने पर उस घर से उस सामान को प्राप्त करके काम में ले लेते हैं, उसी तरह से Memory में भी अलग–अलग Storage Cells में हम अपनी जरूरत के अनुसार अलग–अलग मान Store कर सकते हैं और जरूरत पडने पर उस Data को प्राप्त कर के काम में ले सकते हैं।

एक उदाहरण द्वारा हम Computer की कार्यप्रणाली को समझने की कोशिश करते हैं। जब हम Keyboard (Input Device) पर कोई Key Press करते हैं, तो CPU का Control Unit Active हो जाता है और Input हो रहे Character को Memory में किसी Storage Location पर Store कर देता है।

माना हम Keyboard से एक अंक 6 Input करते हैं, तो CPU का Control Unit इस मान को Memory की किसी Storage Location पर Store कर देता है। अब हम + का चिन्ह Keyboard से Press करते हैं। Control Unit उसे भी Memory की किसी Storage Cell में Store कर देता है। अब हम 4 Input करते हैं। Control Unit इसे भी Memory में Store करता है। माना कि ये तीनों मान निम्नानुसार Memory की विभिन्न Storage locations पर Save हो रही हैं –

1001	1002	1003	1004	1005	1006	1007	1008	1009	1010
6		+			4			1	0

अब यदि Input किए गए दोनों अंकों को जोडना हो तो Control Unit Input किए गए पहले मान को Storage Cell 1001 से उठाता है और उसे Binary Format में Convert करके CPU के Registers में भेज देता है। फिर इस अंक पर Processing करने के लिए Control Unit को Memory Location 1003 से उठाता है और CPU को इस अंक को जोडने की प्रक्रिया करने की Instruction देता है।

फिर Control Unit दूसरे मान 4 को Binary Format में Convert करके CPU के Register में भेज देता है और ALU को इन दोनों अंको को जोडने की सूचना देता है। ALU Registers से इन Data को प्राप्त करके उन्हें जोडता है और जोडे गए मान को CPU के Registers में Store कर देता है। यहां से वापस Control Unit Registers में Stored मान को Memory में Storage Cell 1009 व 1010 पर Store कर देता है। इस तरह से दो संख्याओं को जोडने की प्रक्रिया पूरी होती है।

Computer एक Digital Machine है। Computer तभी कोई काम कर सकता है जब उसे किसी काम को करने के लिए Program किया गया हो। Programming दो तरह की होती है। पहली Programming वह होती है जो किसी Computer को काम करने लायक अवस्था में लाने के लिए की जाती है। इस Programming को भी दो भागों में बांटा जा सकता है:

Hardware Programming

इस Programming के अन्तर्गत Computer के Hardware यानी Computer के Motherboard पर लगाए गए विभिन्न प्रकार के Chips व Computer से जुडे हुए अन्य विभिन्न प्रकार के Peripherals जैसे कि Keyboard, Mouse, Speaker, Monitor, Hard Disk, Floppy Disk, CD Drive आदि को Check करने व Control करने के लिए हर Mother Board पर एक BIOS Chip लगाई जाती है। इस BIOS Chip का मुख्य काम Computer को ON करते ही विभिन्न प्रकार के Devices को Check करना होता है।

यदि Computer के साथ जुडी हुई कोई Device ढंग से काम नहीं कर रही है, तो BIOS User को विभिन्न प्रकार की Error Messages देता है। BIOS Chip के अन्दर ही प्रोग्राम को लिखने का काम BIOS बनाने वाली Company करती है। इसे Hard Core Programming या Firmware कहा जाता है। Hardware Programming में Chip को बनाते समय ही उसमें Programming कर दी जाती है। किसी भी Computer के Motherboard पर लगी BIOS Chip यदि खराब हो जाए, तो Computer किसी भी हालत में काम करने लायक अवस्था में नहीं आ सकता यानी Computer कभी Boot नहीं होता।

Software Programming

Computer को काम करने लायक अवस्था में लाने के लिए जिस Software को बनाया जाता है, उसे Operating System Software कहा जाता है। BIOS Chip का काम पूरा होने के बाद Computer का पूरा Control Operating System Software के पास आ जाता है। Computer के पास BIOS से Controlling आने के बाद सबसे पहले Memory में Load होने वाला Software Operating System Software ही होता है। इसे Master Software या Operating System Software भी कहते हैं।

आज विभिन्न प्रकार के Operating System Software बन चुके हैं जैसे DOS, Windows, OS/2, WRAP, Unix, Linux आदि। इन सभी Software का मुख्य काम Computer को Boot करके User के काम करने योग्य अवस्था में लाना होता है।

दूसरी Programming वह Programming होती है, जिससे Computer हमारी बात को समझता है और हमारी इच्छानुसार काम करके हमें परिणाम प्रदान करता है। इन्हें Application Software कहा जाता है। हम किसी भी Operating System के लिए किसी भी भाषा में जब कोई Program लिखते हैं, तो वास्तव में हम Application Software ही लिख रहे होते हैं।

Application Software का मुख्य काम किसी विशेष समस्या का समाधान प्रदान करना होता है। MS-Office, Corel-Draw, PageMaker, PhotoShop आदि Application Software के उदाहरण हैं, जो हमें किसी विशेष समस्या का समाधान प्रदान करते हैं। जैसे यदि हमें Photo

Editing से सम्बंधित कोई काम करना हो तो हम PhotoShop जैसे किसी Application Software को उपयोग में लेते हैं।

Language

भाषा, दो व्यक्तियों के बीच संवाद, भावनाओं या विचारों के आदान–प्रदान का माध्यम प्रदान करती है। हम लोगों तक अपने विचार पहुंचा सकें व अन्य लोगों के विचारों का लाभ प्राप्त कर सकें, इसके लिये जरूरी है कि संवाद स्थापित करने वाले दोनों व्यक्तियों के बीच संवाद का माध्यम समान हो। यही संवाद का माध्यम भाषा कहलाती है। अलग–अलग स्थान, राज्य, देश, परिस्थितियों के अनुसार भाषा भी बदलती रहती हैं, लेकिन सभी भाषाओं का मकसद संदेशों या सूचनाओं का आदान प्रदान करना ही होता है।

ठीक इसी तरह कम्प्यूटर की भी अपनी कई भाषाएं है, जो जरूरत व उपयोग के अनुसार विकसित की गई है। हम जानते हैं, कि कम्प्यूटर एक इलेक्ट्रॉनिक मशीन मात्र है। ये हम सजीवों की तरह सोंच विचार नहीं कर सकता है और ना ही हमारी तरह इनकी अपनी कोई भाषा है, जिससे हम इनसे सम्बंध बना कर सूचनाओं का लेन–देन कर सकें।

इसलिये कम्प्यूटर को उपयोग में लेने के लिये एक ऐसी भाषा की जरूरत होती है, जिससे हम हमारी भाषा में कम्प्यूटर को सूचनाएं दें व कम्प्यूटर उसे उसकी मशीनी भाषा में समझे और हमारी चाही गई सूचना या परिणाम को हमें हमारी भाषा में दे ताकि हम उसे हमारी भाषा में समझ सकें।

Java एक ऐसी ही भाषा है। ये एक High Level Language है। इस Language में हम सामान्य English के शब्दों में Code लिखते हैं। इन Codes को Java का एक Software Program जिसे Compiler कहते हैं, उस भाषा में Convert करता है, जिसे Computer समझ सकता है। इस Software की मदद से Compute हमारे लिखे गए Codes के अनुसार काम करता है और हमें वह परिणाम प्रदान करता है, जो हम Computer से चाहते हैं।

Java का Program लिखने के लिए एक Simple Word Processor की जरूरत होती है। Windows के साथ आने वाले Notepad को हम Java का Program लिखने के लिए Use कर सकते हैं। Java में हर Program को एक Class के रूप में लिखा जाता है। Program लिखने के बाद Source File को .Java Extension के साथ Save किया जाता है।

Java में दो तरह के Programs सबसे ज्यादा लिखे जाते हैं। जब हम Java Application लिखते हैं तो उस Application में हमेंशा एक **main()** Method होता है। Java एक Case Sensitive Language है इसलिए Capital Letters में लिखा गया नाम व Small Letters में लिखा गया नाम दोनों नाम अलग–अलग होते हैं।

Java Compiler (javac)

Java Compiler (Javac) Java Developer's Kit का एक Component है, जिसका प्रयोग Java की Source Code File को Bytecodes Executable File में Convert करने के लिए किया जाता है, ताकि वह File Java Runtime Environment (JRE) System में Run हो सके। Java की Source Code File का Extension .java होता है।

Java की Source Code File एक Standard ASCII Text File होती है। ये Java के Compiler का काम होता है कि वह Java Source Code File को Process करे और Executable Java Bytecodes Class File Create करे। Executable Bytecodes Class File का Extension .class होता है और ये अपनी Useable Form में Java Class को Represent करते हैं।

Java Compiler हमारी Source File की हर Class के लिए एक .Class File Generate करता है। तकनीकी रूप से हम एक ही Source File में एक से अधिक Class Define कर सकते हैं लेकिन Compiler एक ही Source File में Define की गई एक से अधिक Classes के लिए भी अलग–अलग **.class** File Create करता है।

Java Compilers इस बात के लिए जिम्मेदार होते हैं कि एक Java Source File से Java Executable Bytecodes File किस प्रकार से Generate की जाए जो कि Java Runtime System में Run हो सकें और Java Virtual Machine, जो कि Java Runtime System का एक हिस्सा है, इस बात के लिए जिम्मेदार होता है, कि Bytecodes को किस प्रकार से Interpret किया जाए।

Java Compiler एक Command Line Tool है। इसका मतलब ये हुआ कि इन्हें DOS Prompt पर ही Execute किया जा सकता है। इनको Use करने का Syntax निम्नानुसार होता हैः

```
Command Prompt > javac Options Filename
```

इस Command में Filename Argument उस Java Source Code File का नाम होता है जिसे Compile करना है। इस File में जितनी भी Classes Define की गई होती हैं, उन सभी की Bytecodes वाली .class File बनती है। ये Compiler उन सभी Classes को Bytecodes Class File में Convert कर देता है जो एक दूसरे पर Depend होती हैं।

यानी मानलो कि एक File *X.Java* है और इस File में एक दूसरी File जिसका नाम *B.Java* है, की Class को Derive किया गया है। तो इस स्थिति में हम यदि *B.Java* को Compile करते हैं तो Compiler *X.Java* की भी Compiling करेगा और दोनों फाइलों के सभी Classes की Bytecodes (.Class) File Generate करेगा।

How to Get Complete PDF EBook

आप **Online Order** करके **Online** या **Offline** Payment करते हुए इस Complete EBook को तुरन्त Download कर सकते हैं।

Order करने और पुस्तक को Online/Offline Payment करते हुए खरीदने की पूरी प्रक्रिया की विस्तृत जानकारी प्राप्त करने के लिए आप **BccFalna.com** के निम्न Menu Options को Check Visit कर सकते हैं।

How to Make Order

How to Order?

How to Buy Online

How to Pay Online using PayUMoney

How to Pay Online using Instamojo

How to Pay Online using CCAvenue

How to Buy Offline

How to Pay Offline

Bank A/c Details

जबिक हमारे Old Buyers के **Reviews** भी देख सकते हैं ताकि आप इस बात का निर्णय ले सकें कि हमारे Buyers हमारे PDF EBooks से कितने Satisfied हैं और यदि आप एक से अधिक EBooks खरीदते हैं, तो **Extra Discount** की Details भी Menubar से प्राप्त कर सकते हैं।