CST205: OBJECT ORIENTED DESIGN AND PROGRAMMING USING JAVA

B.Tech. CSE

Semester III

Viswajyothi College of Engineering and Technology

Course Objectives

1. Write Java programs using the object oriented concepts - classes, objects, constructors, data hiding, inheritance and polymorphism
2. Utilise datatypes, operators, control statements, built in packages & interfaces, Input/ Output Streams and Files in Java to develop programs
3. Illustrate how robust programs can be written in Java using exception handling mechanism
4. Write application programs in Java using multithreading and database connectivity.
5. Write Graphical User Interface based application programs by utilising event handling features and Swing in Java

- Text Books: Reference Books:
- **1. Herbert Schildt, Java:The Complete Reference, 8/e,Tata McGraw Hill, 2011.**
- 2. Rajib Mall, Fundamentals of Software Engineering, 4th edition, PHI, 2014.
- 3. Paul Deitel, Harvey Deitel, Java How to Program, Early Objects 11th Edition, Pearson, 2018.

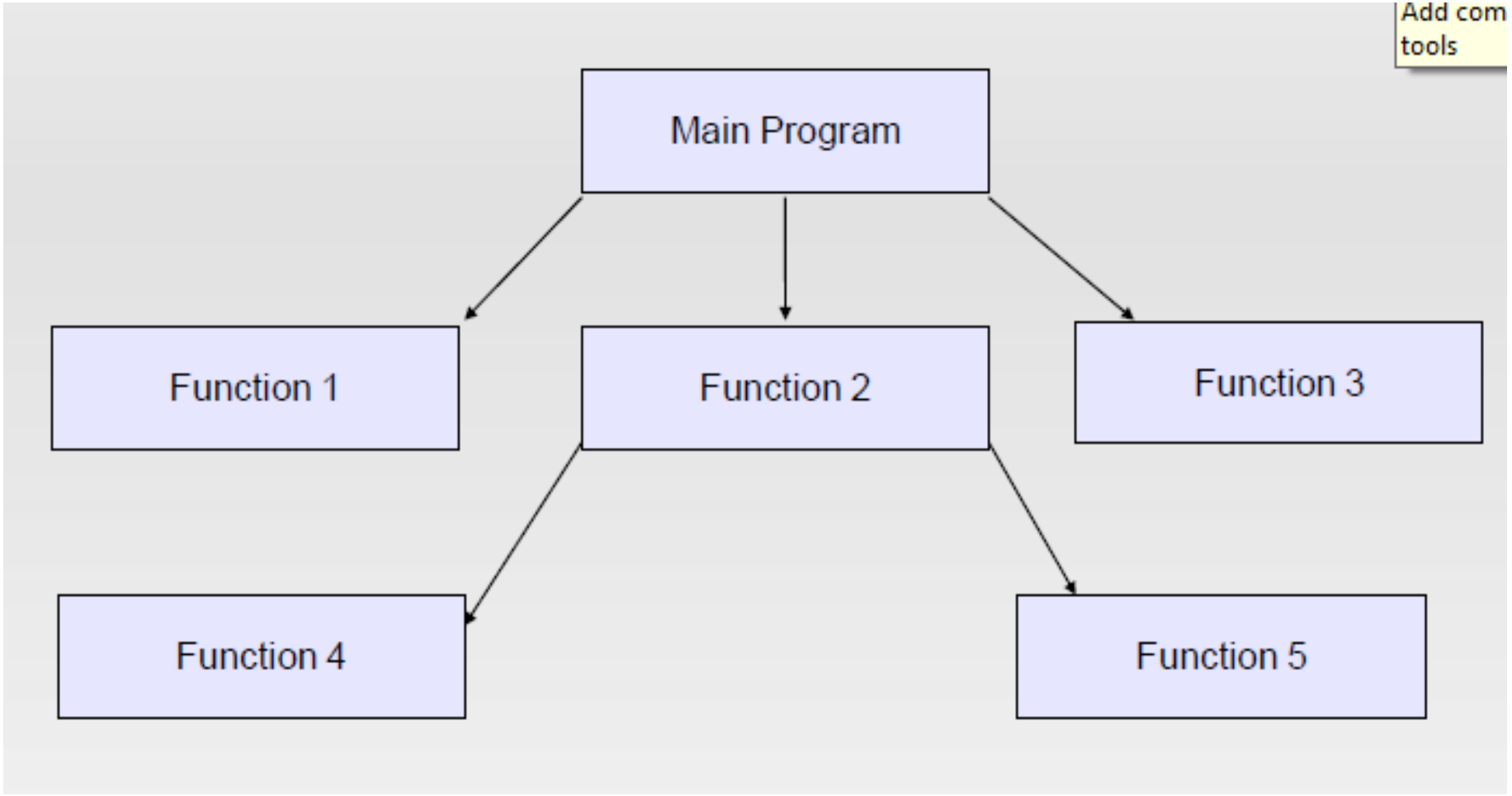
Syllabus of Module 1

- **Introduction:** Approaches to Software Design - Functional Oriented Design, Object Oriented Design, Case Study of Automated Fire Alarm System.
- Object Modeling Using Unified Modeling Language (UML) – Basic Object Oriented concepts, UML diagrams, Use case model, Class diagram, Interaction diagram, Activity diagram, State chart diagram.
- Introduction to Java - Java programming Environment and Runtime Environment, Development Platforms -Standard, Enterprise. Java Virtual Machine (JVM), Java compiler, Bytecode, Java applet, Java Buzzwords, Java program structure, Comments, Garbage Collection, Lexical Issues.

Function-Oriented Design

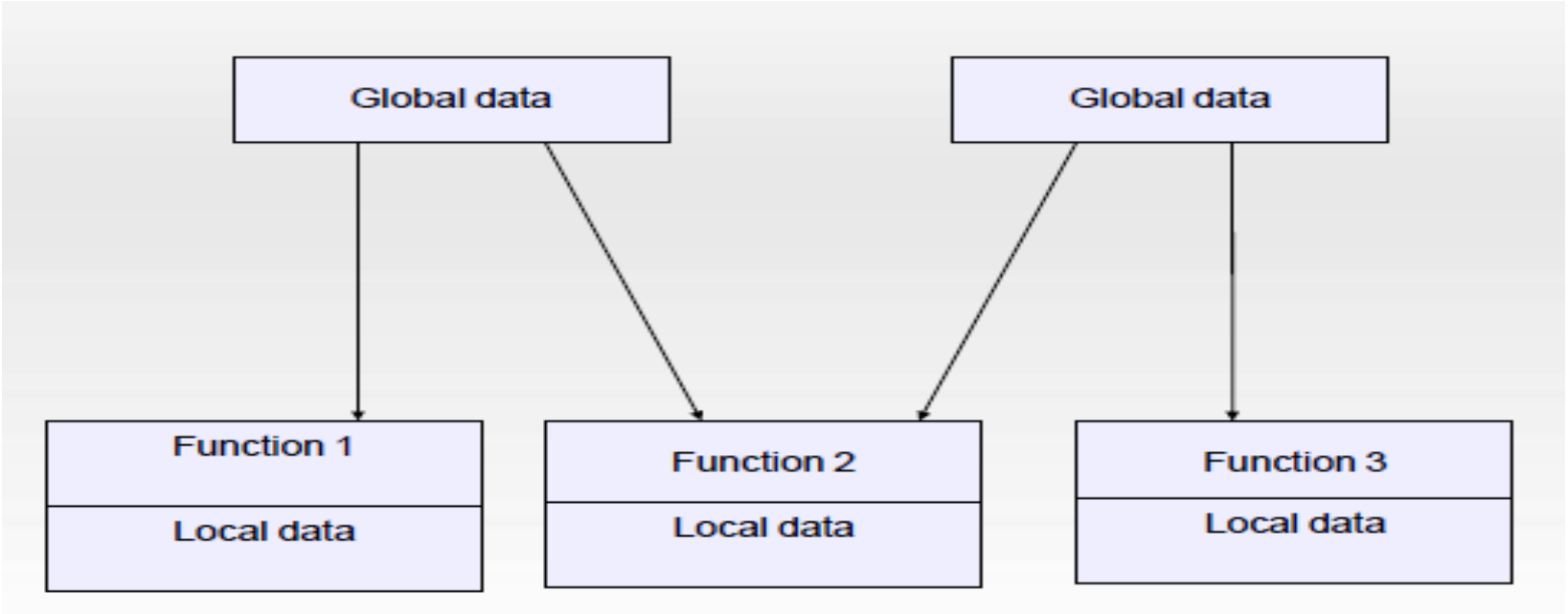
- Function oriented design is the result of focusing attention to the function of the program. This is based on the stepwise refinement which follows top down strategy.
- We start with a high level description of what the program does. Then, in each step, we take one part of our high level description and refine it.
- The refinement of each module is done until we reach the statement level of our programming language.

- Structure of function-oriented programming



Drawback:

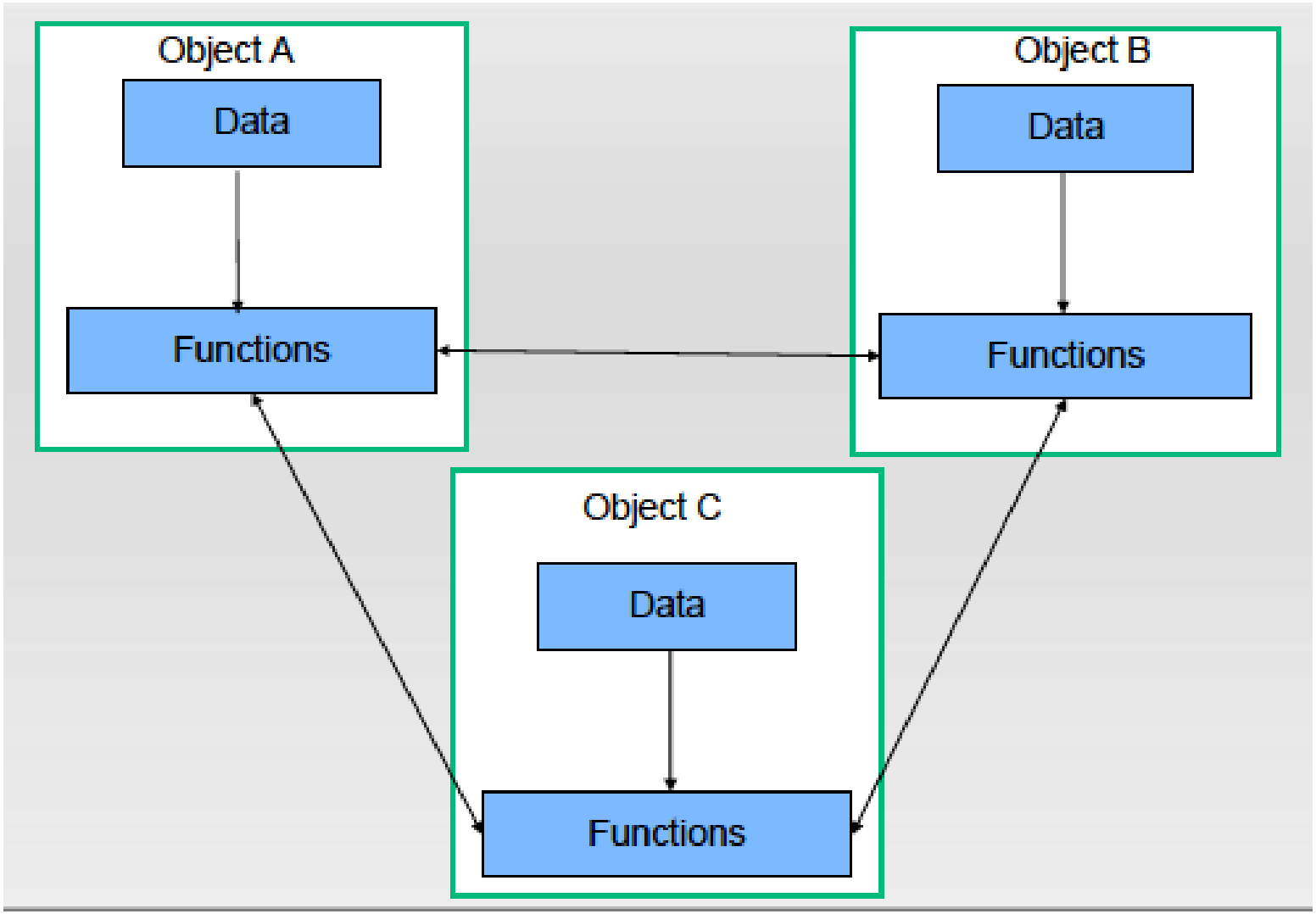
- More importance to functions and very little attention to data that are being used by the functions.
- In multi-function program many important data items are placed as global so that they may be accessed by all the functions. Each function may have its own local data.
- In large program it is very difficult to identify what data is used by which function.



Object-Oriented Programming

- OOP treats data as a critical element in the program development and does not allow it to flow freely around the system.
- It ties data more closely to the functions that operate on it, and protects it from accidental modification from outside functions.
- OOP allows decomposition of a problem into a number of entities called objects.

- **Organization of data & function in OOP**



Features of OOP

- Emphasis is on data rather than procedure or function.
- Programs are divided into what are known as objects.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can be easily added whenever necessary.

Object Oriented Concepts

- Class
- Objects
- Data Abstraction
- Data Encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Message Passing
- The three pillars of object-oriented development:
  - Encapsulation, Inheritance, and Polymorphism.

Classes

- Classes are user-defined data types and behave like the built-in types of a programming language.

- Objects are variable of type class.
- Once a class has been defined we can create any number of objects belonging to that class.

- A class is thus a collection of objects of similar type.

- Syntax used to create an object is no different than the syntax used to create an integer object in C.
- Suppose Dog is a class defined.

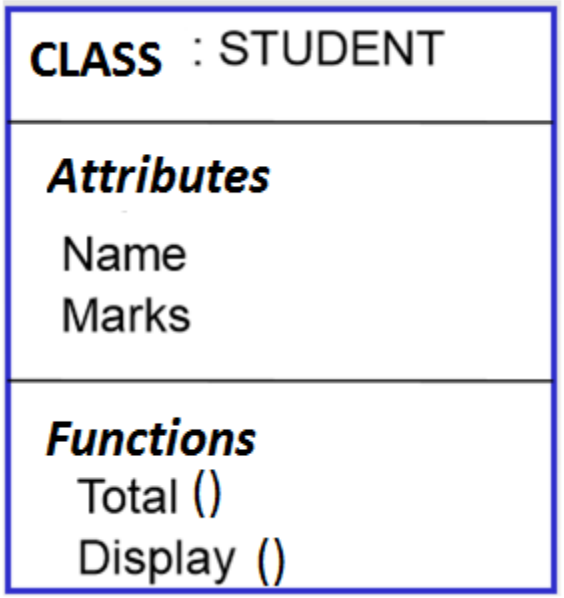
  Dog d1; //Here Dog is Class and d1 is an object of Dog Class.

Objects

- Objects are the basic run-time entities in an object oriented system.
- They may represent a person, a place, a bank account, a table of data or any item that the program must handle.
- Program objects should be chosen such that they match closely with the real-world objects.
- Objects take up space in the memory.
- Each object is said to be an instance of its class.

- Class is composed of attributes that defines the properties of object and functions that display the behaviour .
- Each instance of the class(ie object) has its own value for each attribute but shares the same attribute names and operations with other instances of the class.

- Representation of class



Press Esc to exit full screen

CLASS

```
public class Dog {
    String breed;
    int age;
    String color;

    void bark() {
        int x;
    }

    void run() {
        String s;
    }

    // .. More Methods
}
```

OBJECT

```
Dog dog1 = new Dog();
dog1.breed    = pug;
dog1.age      = 10;
dog1.color    = black;

Dog dog2 = new Dog();
dog2.breed    = labrador;
dog2.age      = 9;
dog2.color    = black;
```

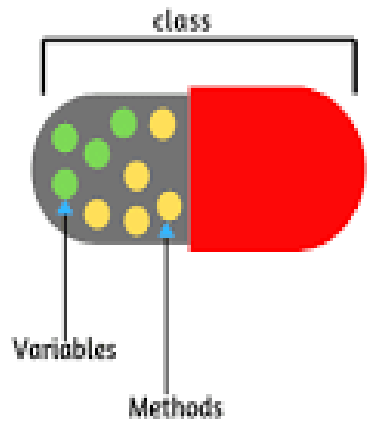
Encapsulation

- The wrapping up of data and functions into a single unit is known as Encapsulation.
- The data is not accessible to the outside world and only those functions which are wrapped in the class can access it.
- This insulation of the data from direct access by the program is called data hiding or information hiding.

```
class
{

  data members
        +
  methods (behavior)

}
```



Inheritance:

- Inheritance is the process by which objects of one class acquire the properties of objects of another class.

- The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived.

- Parent class is called base/superclass. Derived class is called subclass/derived class.

- In OOP the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it.

Inheritance

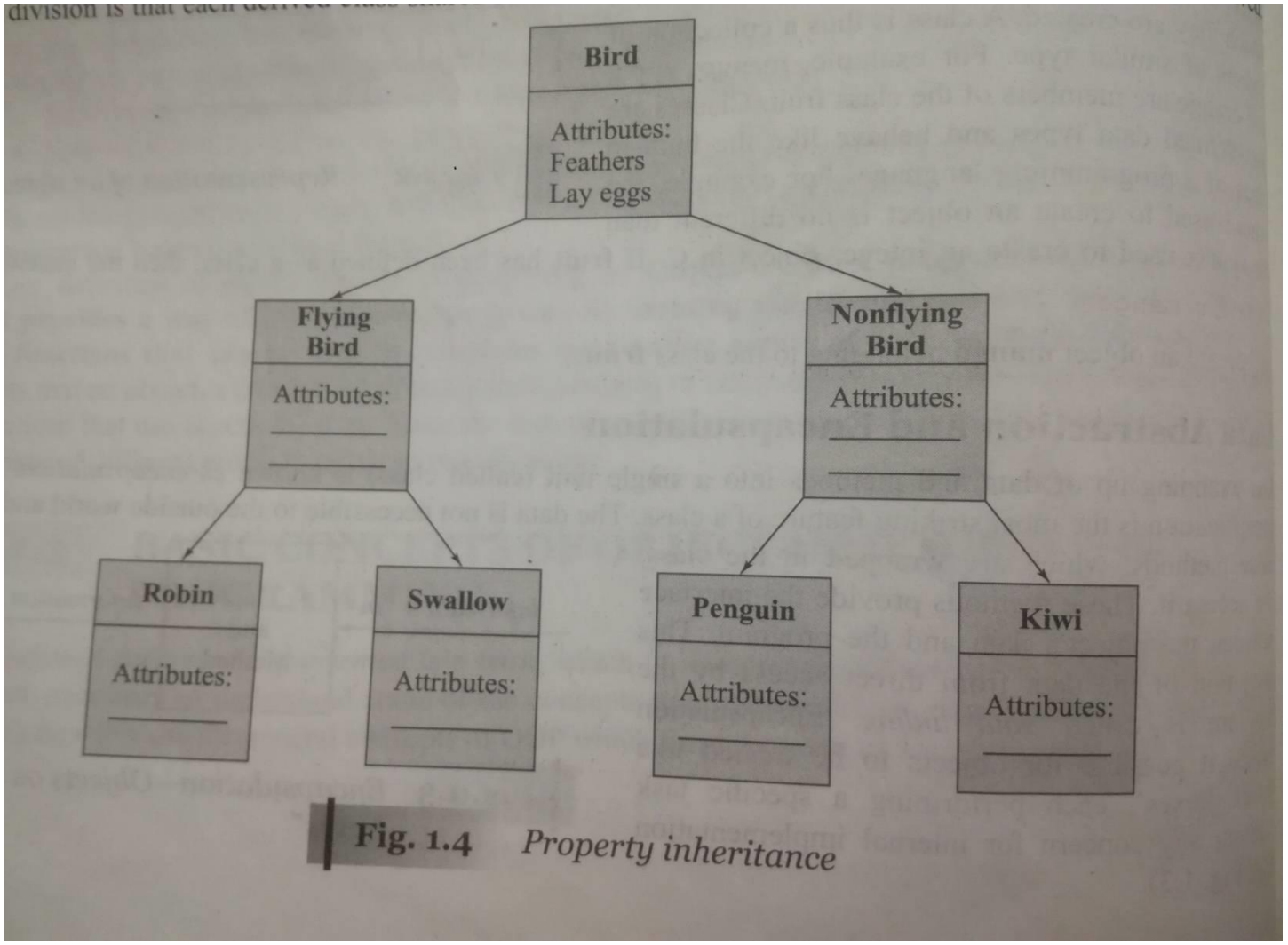


Fig. 1.4 *Property inheritance*

Data Abstraction

- Abstraction refers to the act of representing essential features without including the background details or explanations.

- Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes.

- Since the classes use the concept of data abstraction, they are known as Abstract Data Types.

- Abstract classes have no instances but define common behaviors that can be inherited by more specific classes.

Polymorphism:

- Polymorphism means the ability to take more than one form.
- Different behaviours is exhibited in different instances. The behaviour depends upon the types of data used in the operation.
- Example: Operation of addition.
  - For two numbers the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation.

- The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.

- Using a single function name to perform different types of tasks is known as function overloading.

Polymorphism

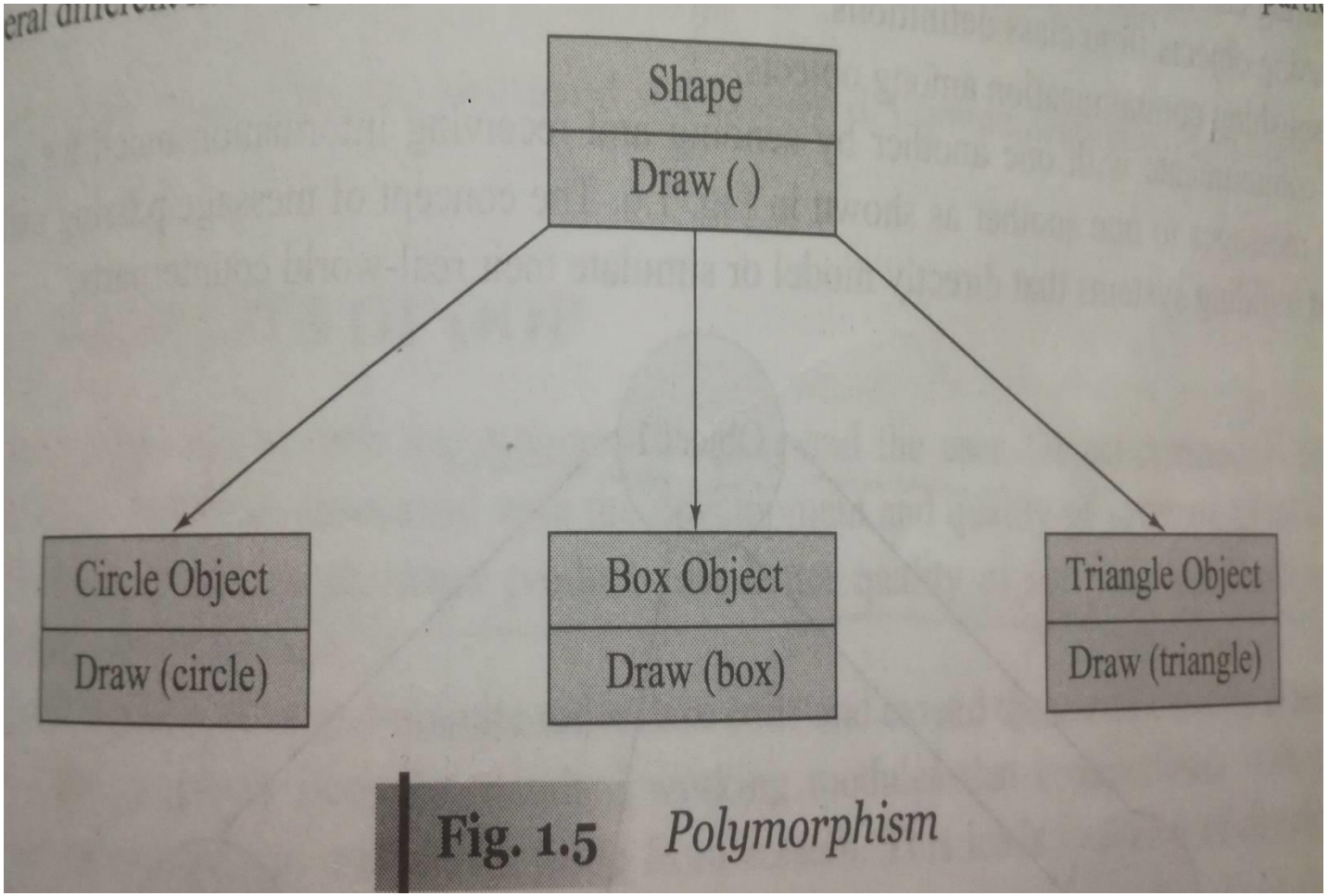


Fig. 1.5 *Polymorphism*

Dynamic Binding

- The process of determining at run time which function to invoke is termed as dynamic binding.
- The code associated with a given procedure call is not known until the time of the call at run-time.
- It is associated with polymorphism and inheritance.

Message Communication:

- An object-oriented program consist of a set of objects that communicate with each other.

- Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.
  - Eg: emp.salary(emp_name); Here emp is the object, salary is the message and emp_name is the information

Advantages of OOP

- Redundant code is eliminated and existing classes can be extended by the principle of inheritance.
- The principle data hiding helps the programmer to build secure programs.
- It is possible to have multiple objects to coexist without any interference.
- Object-oriented systems can be easily upgraded from small to large systems.

Introduction to Java

Java's History

- Originally developed by James Gosling at **Sun Microsystems** (which is now a part of Oracle Corporation)
- Originally called Oak
- Released in 1995
- Aimed at producing an operating environment for networked devices and embedded systems.

- Design objectives for the language
  - Simple, secure, object-oriented, architecture-neutral and portable, distributed and dynamic.

Java Programming Environment

Consists of

- Java language – used by programmers to write the application.

- The Java Virtual Machine(JVM)-used to execute the application.

Java Runtime Environment(JRE)

- It is what you get when you download java software.

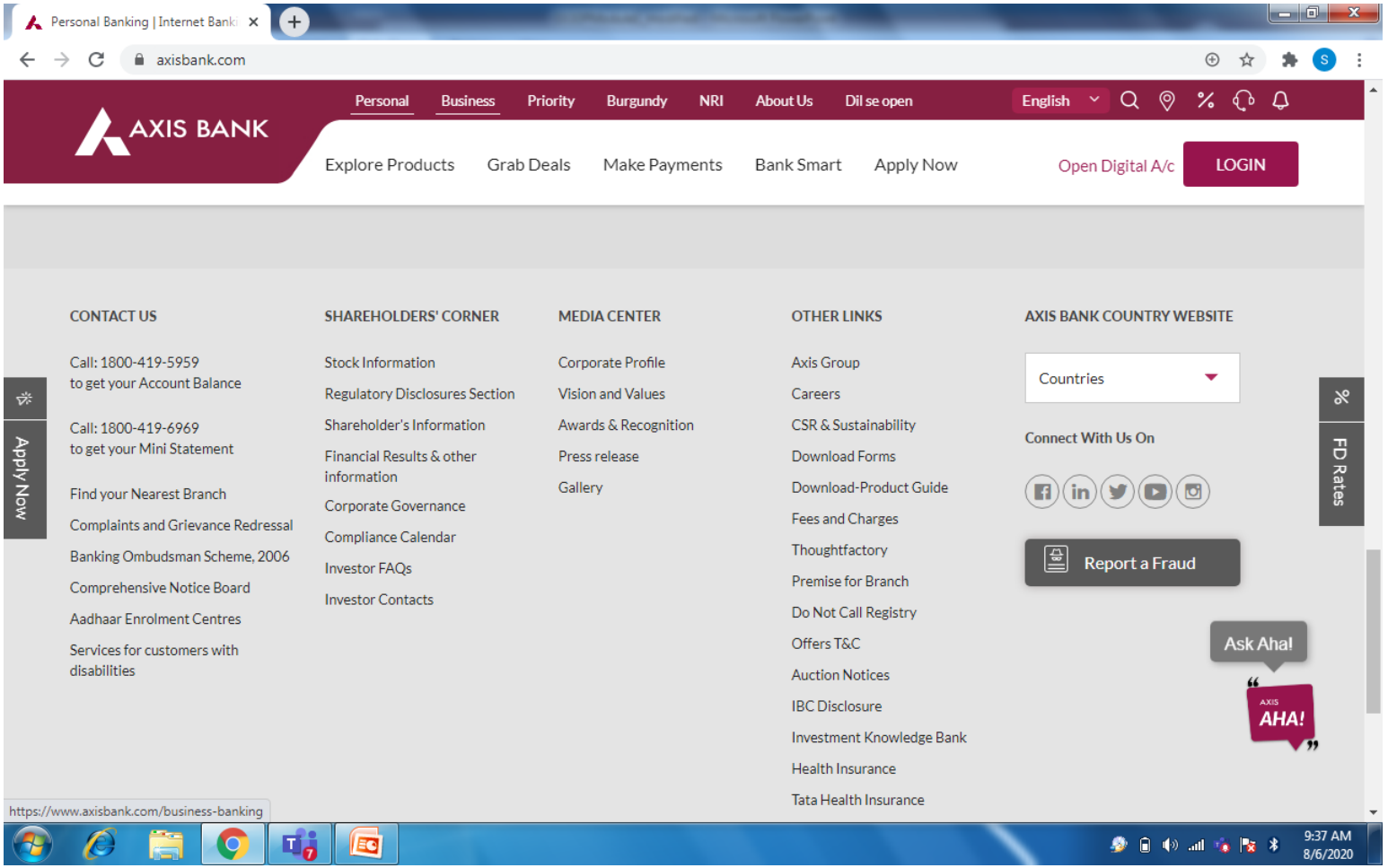
- JRE consists of Java Virtual Machine(JVM), Java platform core classes and supporting java platform libraries.

Java Programming Language Platforms

- A Java platform is a particular environment in which Java programming language applications run.

- All Java platforms consist of a Java Compiler , Java Virtual Machine (JVM) and an application programming interface (API).
  - An ***API*** is a collection of software components(classes, packages etc) that we can use to create other software components or applications. It is an intermediary software that allows the applications to talk to each other.
  - The API is a collection of available java classes, packages and interfaces.

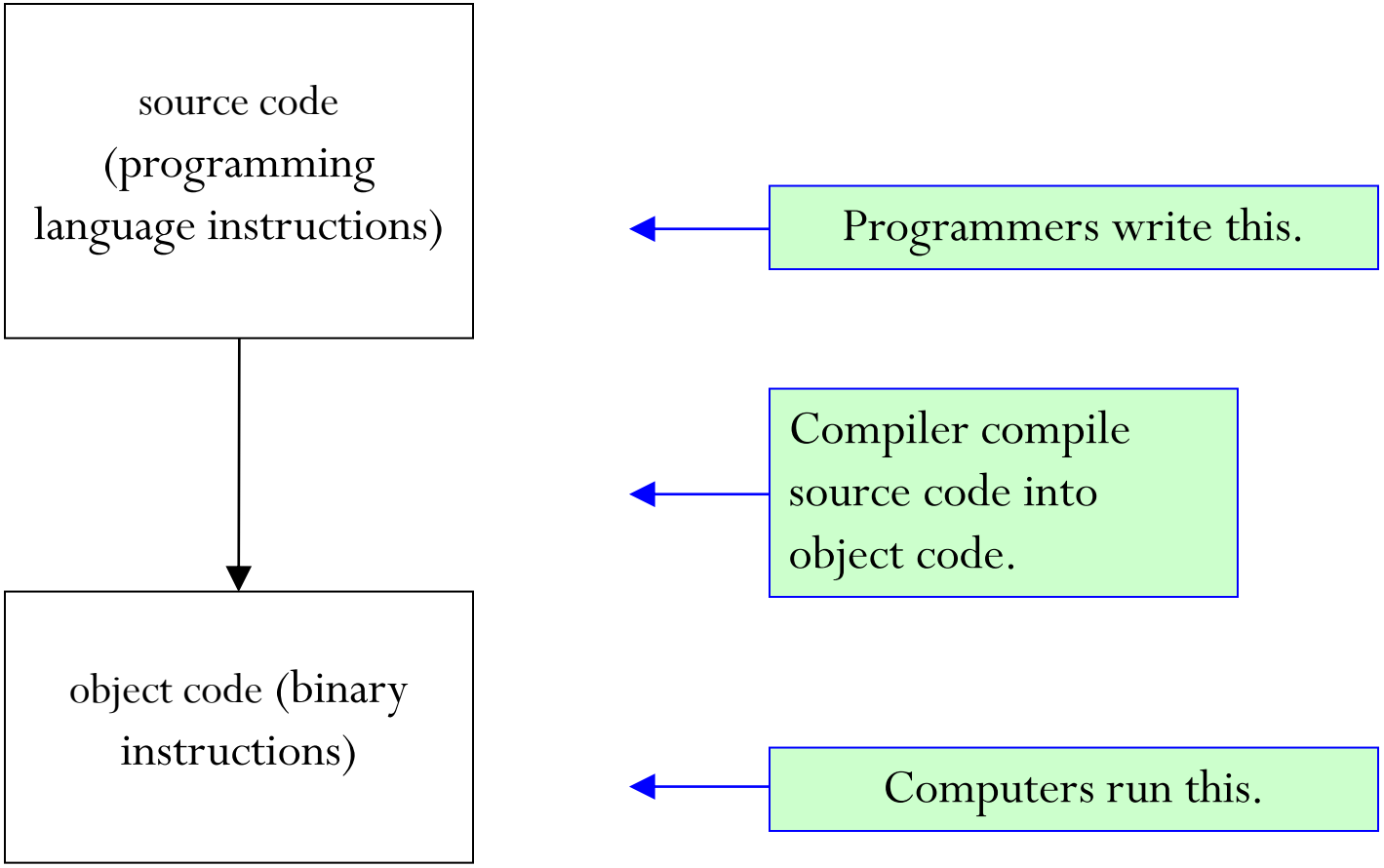
W


Personal Banking | Internet Banki
axisbank.com
AXIS BANK
Personal
Business
Priority
Burgundy
NRI
About Us
Dil se open
English
Explore Products
Grab Deals
Make Payments
Bank Smart
Apply Now
Open Digital A/c
LOGIN
CONTACT US
Call: 1800-419-5959
to get your Account Balance
Call: 1800-419-6969
to get your Mini Statement
Find your Nearest Branch
Complaints and Grievance Redressal
Banking Ombudsman Scheme, 2006
Comprehensive Notice Board
Aadhaar Enrolment Centres
Services for customers with disabilities
SHAREHOLDERS' CORNER
Stock Information
Regulatory Disclosures Section
Shareholder's Information
Financial Results & other information
Corporate Governance
Compliance Calendar
Investor FAQs
Investor Contacts
MEDIA CENTER
Corporate Profile
Vision and Values
Awards & Recognition
Press release
Gallery
OTHER LINKS
Axis Group
Careers
CSR & Sustainability
Download Forms
Download-Product Guide
Fees and Charges
Thoughtfactory
Premise for Branch
Do Not Call Registry
Offers T&C
Auction Notices
IBC Disclosure
Investment Knowledge Bank
Health Insurance
Tata Health Insurance
AXIS BANK COUNTRY WEBSITE
Countries
Connect With Us On
Report a Fraud
Ask Aha!
AXIS AHA!
Apply Now
FD Rates
https://www.axisbank.com/business-banking
9:37 AM
8/6/2020

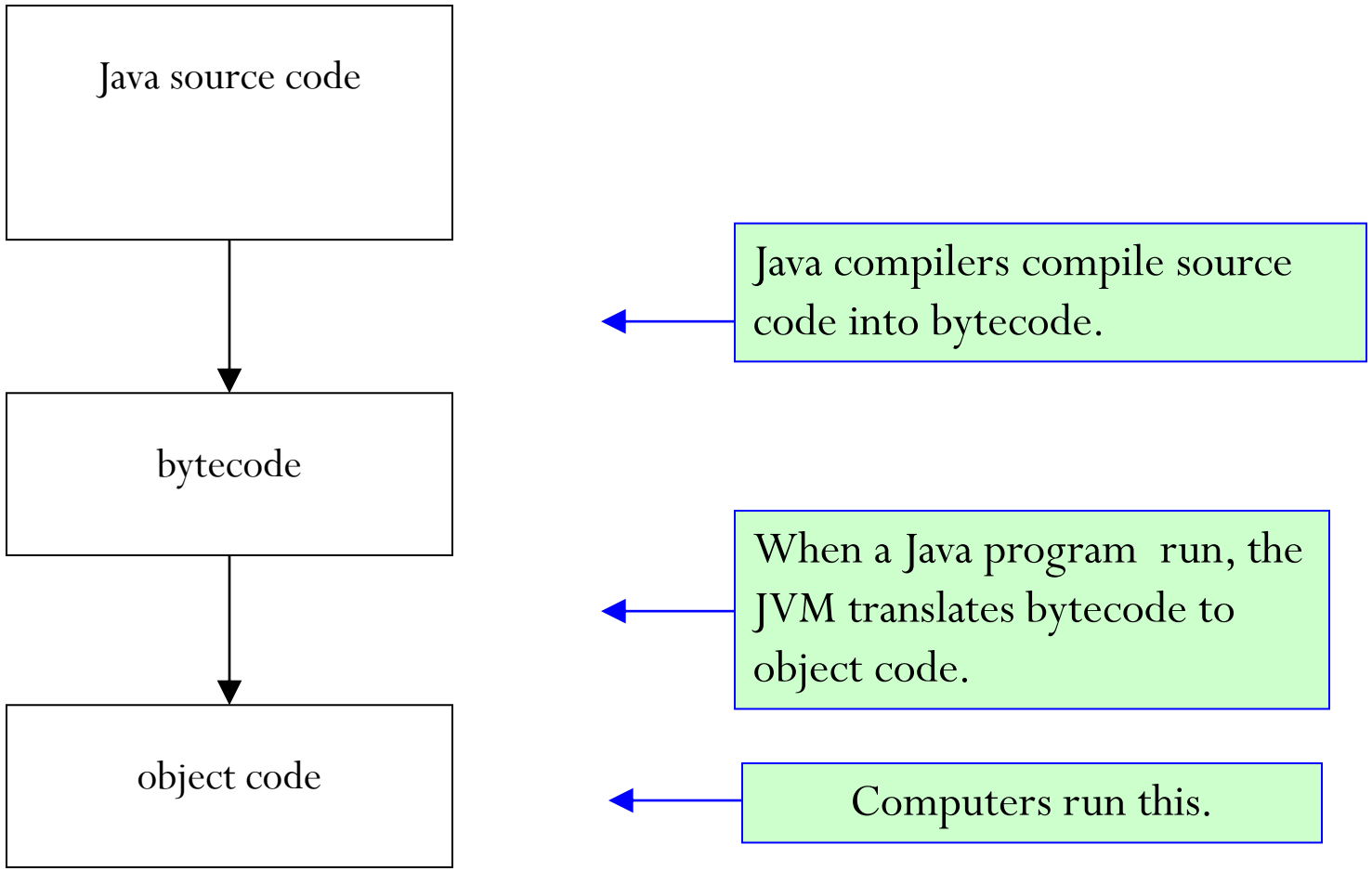
Java Programming Language Platforms

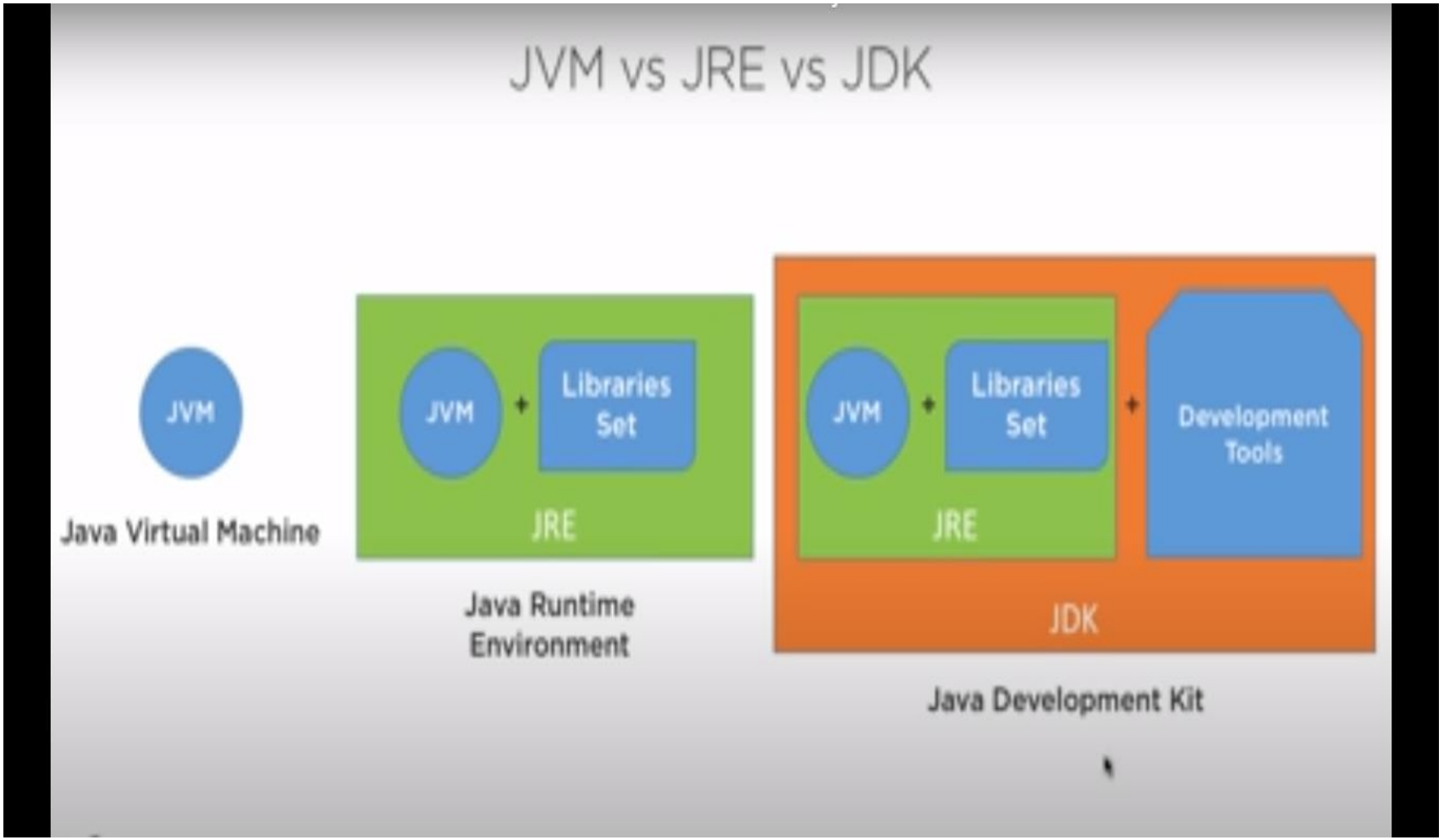
- Java Platform, Standard Edition (J2SE) or Java SE
  - Java SE's API provides the core functionality of the Java programming language.
  - J2SE can be used to develop standalone applications or applets.
- Java Platform, Enterprise Edition (J2EE) or Java EE
  - The Java EE platform provides an API and runtime environment for developing and running large-scale applications.
- Java platform Micro Edition (J2ME) or Java ME
  - J2ME can be used to develop applications for mobile devices.
- JavaFX
  - JavaFX is a platform for creating rich internet applications

The Compilation Process for Non-Java Programs



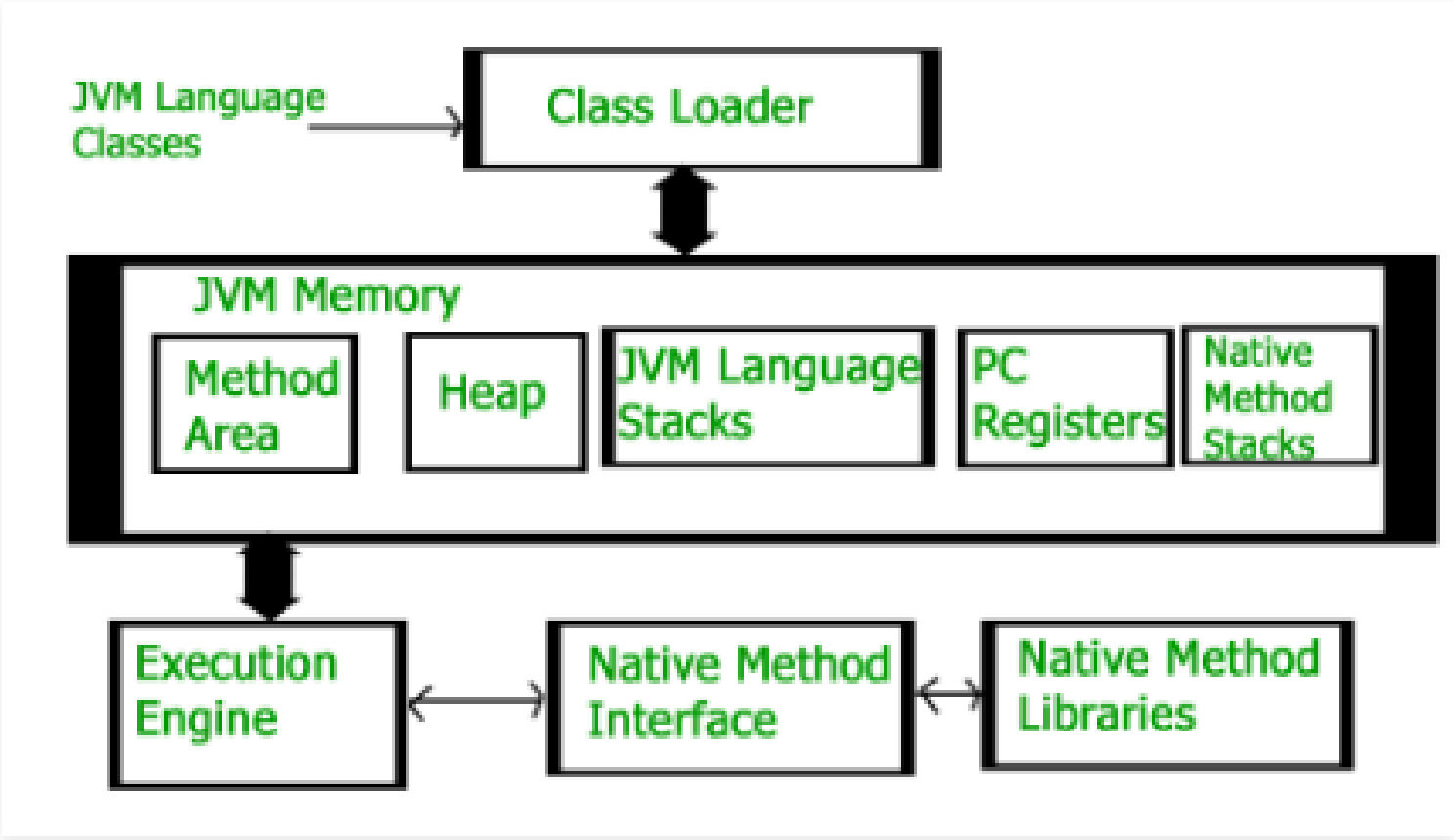
The Compilation Process for Java Programs




JVM vs JRE vs JDK
JVM
Java Virtual Machine
JVM
Libraries Set
JRE
Java Runtime Environment
JVM
Libraries Set
Development Tools
JRE
JDK
Java Development Kit

Java Virtual Machine (JVM)

- *Bytecode* is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the ***Java Virtual Machine* (*JVM*).**
- It is a platform-independent execution environment that converts Java bytecode into machine language and executes it.
- As a Java program's bytecode runs, the bytecode is translated into object code by the computer's bytecode interpreter program. Called *Java Virtual Machine*.
- JVM architecture in Java contains classloader, memory area, execution engine etc.
  - The class loader is a subsystem used for loading class files. It performs three major functions viz. Loading, Linking, and Initialization.


JVM Language Classes
Class Loader
JVM Memory
Method Area
Heap
JVM Language Stacks
PC Registers
Native Method Stacks
Execution Engine
Native Method Interface
Native Method Libraries

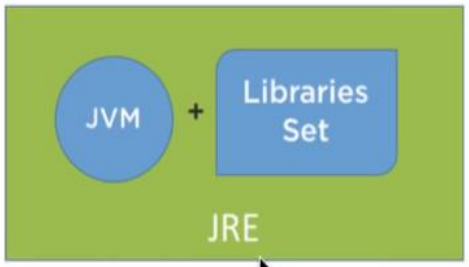
Java Virtual Environment (JVM):
It is an abstract machine and does not have a physical existence.

The aim of JVM is to run Java programs irrespective of the OS, making it platform independent.
Tasks of JVM:

- Load and verify the code
- Execute the code
- Provides a run-time environment to execute Java bytecode.

First, the Java compiler compiles the source code into bytecode. Then the JVM executes the bytecode. Thus, JVM converts the Java bytecode into machine code.

JRE



Java Runtime Environment

JRE → JVM + Set of Libraries

It is the implementation of JVM

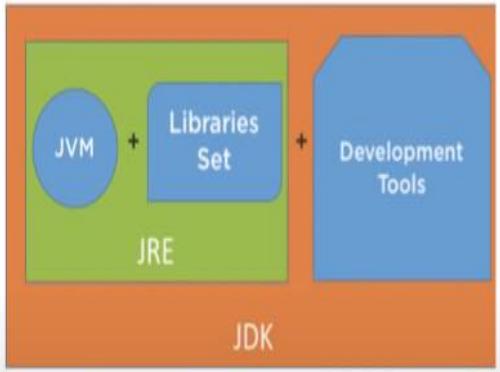
To run any Java code JRE is minimum required.

JRE contains set of libraries that JVM uses at runtime.

JRE physically exists.

JRE is platform dependent.

JDK



Java Development Kit

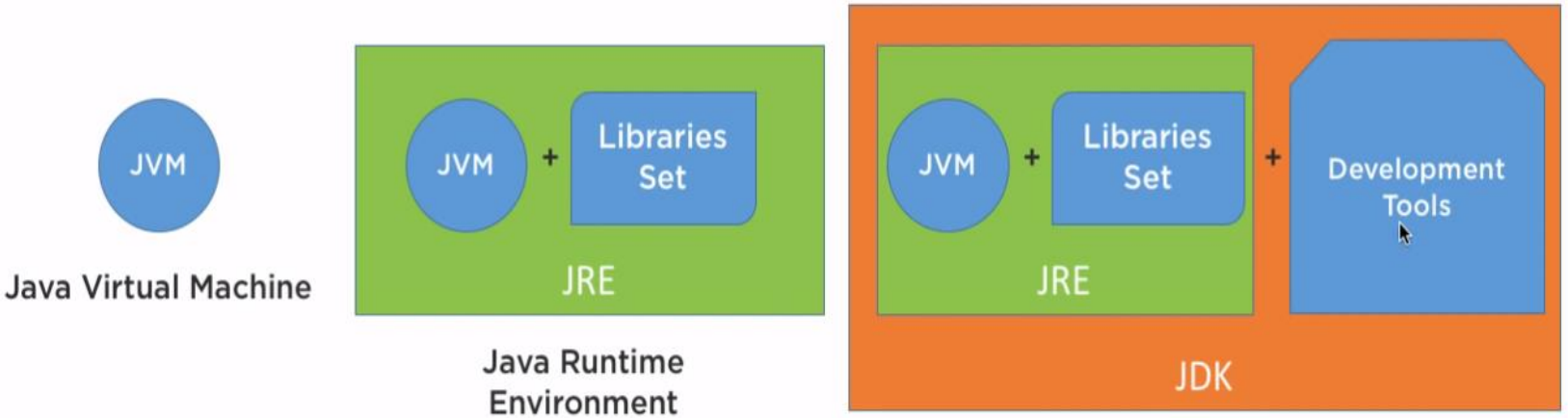
JDK → JRE + Development Tools

It is a full featured Software Development Kit.

It contains JRE + Development Tools

- → JRE
  - JVM + Libraries
- → Development Tools
  - Debugger + Compiler + JavaDoc

Practical Purpose?



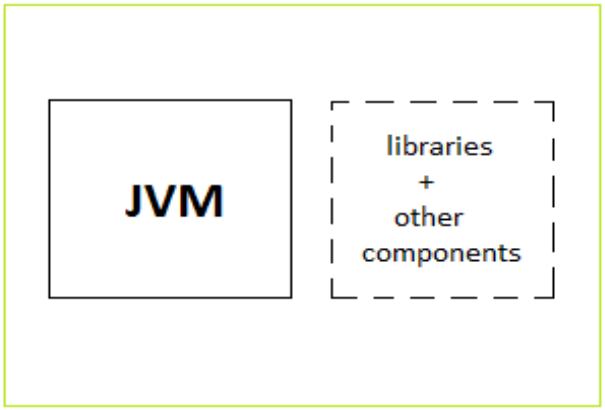
Just provides environment

To run Code

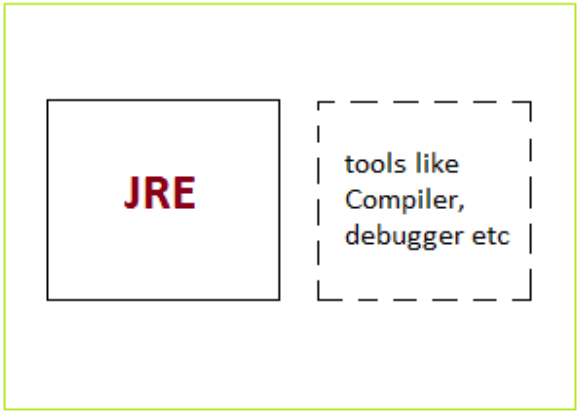
To run Code + Programming

Comparison of JRE and JDK

- **JRE** :
  - The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language.
  - JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications.
- **JDK** :
  - The JDK also called Java Development Kit is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications.



JRE - Java Runtime Environment



JDK - Java Development Kit

Java Applications and Applets

Java can be used to create two types of programs

- An ***application*** is a program that runs on your computer, under the operating system of that computer.

- An ***applet*** is an application designed to be transmitted over the Internet and executed by a Java-compatible Web browser. It is embedded in HTML page using APPLET or OBJECT tag.
  - An applet is a program that can react to user input and dynamically change

Java buzzwords(Characteristics and Features of Java)

- **Java Is Simple**
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral

  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Java inherits the C/C++ syntax and many of the object-oriented features of C++ . So JAVA is easier for programmers to learn it after C++.
- Removed many confusing and rarely-used features

  Eg: explicit pointers, operator overloading
- No need to remove unreferenced objects because there is Automatic Garbage Collection in java

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- **Java Is Secure**
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Every time a user compiles the Java program, the Java compiler creates a class file with Bytecode, which is tested by the JVM at the time of program execution for viruses and other malicious files.
- No explicit pointer so memory blocks cannot be accessed.
- Exception enables Java to capture a series of errors that helps developers to get rid of risk of crashing the system.
- Java's access-control functionality on variables and methods within the objects provide secure program by preventing access to the critical objects from the untrusted code.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- **Java Is Object-Oriented**
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Basic concepts of OOPs are:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation
- Java supports Object Oriented concepts.
- All program code and data reside within objects and classes.
- The object model in Java is simple and easy to extend.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- **Java Is Robust**
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Robustness means strong and reliable.
- Java is reliable because of the following reasons
  - Uses strong memory management i.e. lack of pointers hence avoids security problem
  - There is automatic garbage collection
  - There is exception handling mechanism
  - There is type checking mechanism. Java is a strictly typed language. It checks the code at compile time as well as at run time

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- **Java Is Compiled and Interpreted**
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Java compiler translates source code into bytecode instructions.
- The bytecode is machine-independent and can run on any machine that has a Java interpreter(JVM).
- Java interpreter(JVM) generates machine code that can be directly executed by the machine.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- **Java Is Architecture-Neutral or Platform Independent**
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Java code is compiled by the compiler and converted into bytecode.
- This bytecode is platform independent because it can be run on multiple platforms with a Java Virtual Machine (JVM). i.e. Write Once and Run Anywhere(WORA).

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- **Java Is Portable**
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- Java Is Dynamic

- Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- **Java Is Multithreaded**
- High Performance
- Java Is Distributed
- Java Is Dynamic

- A thread is like a separate program, executing concurrently.
- Allows you to write programs that do many things simultaneously.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- **High Performance**
- Java Is Distributed
- Java Is Dynamic

- The execution speed of Java programs improved significantly due to the introduction of Just-In Time Compilation (JIT)
- They can be run on any platform without being recompiled.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- **Java Is Distributed**
- Java Is Dynamic

- Distributed computing involves several computers working together on a network.
- We can create distributed applications in java. RMI and EJB are used for creating distributed applications.
- Java handles TCP/IP protocols.
- Networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file.

Java buzzwords(Characteristics and Features of Java)

- Java Is Simple
- Java Is Secure
- Java Is Object-Oriented
- Java Is Robust
- Java Is Compiled and Interpreted
- Java Is Architecture-Neutral
  or Platform Independent
- Java Is Portable
- Java Is Multithreaded
- High Performance
- Java Is Distributed
- **Java Is Dynamic**

- Java programs carry with them substantial amount of run time information that is used to verify and resolve accesses to object at runtime.
- Java is capable of dynamically linking in new class libraries, methods, and objects.

Java Comments

- **Why Comments?**
- Comments are pieces of text notes that are added to a program in order to describe something about the program or provide any information.
- Comments make the programs more human readable and easier to understand.
- Any statement written as comments is non-executable.
- The compiler and interpreter ignore these statements while compiling the program.
- These do not affect the flow of the program or the output.

Uses of Comments

- **Code Description:**
- **Information about the Source Code:**
- **Planning and Debugging:**
- **Automatic documentation generation:**

Types of *Comments in Java*:

- **Single line Comment:**

This is used to comment on a single line in a program. The syntax is as follows:

```
//This represents a single line Comment
```

- **Multi-line Comment:**

To comment on multiple lines in a program, we use this type of comment. The syntax is as follows:

```
/*This is multi-line comment statement 1
  This is multi-line comment statement 2
*/
```

- **Documentation Comment:**

We place these doc comments above the methods or classes which we want to document. The JDK Javadoc tool uses these doc comments to automatically prepare documentation of the source code. This comment is very much similar to multi-line comment except for an extra "*".

```
/**This represents documentation comments.
 This is Javadoc
 @author Ipsita
*/
```

```
/**
 *The HelloWorld program implements an application that
 * simply displays "Hello World!" to the standard output.
 *
 * @author Zara Ali
 * @version 1.0
 * @since 2014-03-31 */
public class HelloWorld {
public static void main(String[] args)
{ // Prints Hello, World! on standard output.
        System.out.println("Hello World!");
  }
}
```

Java Garbage Collection

- In java, garbage means unreferenced objects.
- Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.
- To do so, we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.
- Advantage of Garbage Collection
  - It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
  - It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

How can an object be unreferenced?

There are many ways:

- By nulling the reference

  eg) Employee e = **new** Employee();

  e = **null**;

- By assigning a reference to another

  eg) Employee e1=**new** Employee();

  Employee e2=**new** Employee();

  e1=e2; //now the first object referred by e1 is available for garbage collection

- By anonymous object etc.

  eg) **new** Employee();

Lexical issues

- Java programs is a collection of White spaces , Identifiers , comments , Literals , Operators , Separators and Keywords.

  -White Spaces

  Java is a free form language. This means that you do not need to follow any special indentation rules. In java , white spaces is a space , tab or new line.

  -Identifiers

  Identifiers are used for class names , method names and variable names. An identifier may be any descriptive sequence of uppercase and lowercase letters , numbers or the underscore and dollar sign design.

Contd..

-Literals

A constant value in java is created by using a literal representation of it. A literal can be used anywhere a value of its type is allowed.

-Comments

There are 3 types of comment in java. First is single line comment and the second one is multi line comment. The third type of comment is called documentation comment. It is used to produce an HTML file that documents your program. It begins with a/** and ends with a*/.

Contd..

-Separators

There are few symbols in java that are used as separators.The most commonly used separator in java is the **semicolon** ' ; '. some other separators are **Parentheses** '( )' , **Braces** ' {} ' , **Bracket** ' [] ' , **Comma** ' , ' , **Period** ' . ' .

- Java Keywords

There are 49 reserved keywords currently defined in java. These keywords cannot be used as names for a variable , class or method.

Contd..

- The **Keywords** are : abstract , assert , boolean , break , byte , case , catch , char , class , const , continue , default , do , double , else , extends , final , finally , float , for , goto , if , implements , import , instanceof , int interface , long , native , new , package , private , protected , public , return , short , static , strictfp , super , switch , synchronized , this , throw , throws , transient , try , void , volatile, while.

A Simple Java Program – Hello.java

```
/*Hello World, first program in java.
   This program prints Hello World.
*/
import java.io.*;
class Hello
{
    public static void main (String args [])
    {
            System.out.println("Hello
    World\n");
    }  //end main
}//end class
```

o/p: Hello World

- To compile:
  - javac Hello.java
- To run:
  - java Hello

A simple C Program Hello.C

```
/* Hello World, first program in C.
   This program prints Hello World.
*/
#include <stdio.h>
void  main()
{
printf("Hello, World!");
 }
```

o/p: Hello World

public static void main (String args [])

- All Java applications begin execution by calling **main( )**
- main( ) must be declared as **public**, since it must be called by code outside of its class when the program is started.
- The keyword **static** allows main( ) to be called without having to instantiate a particular instance of the class. This is necessary since main( ) is called by the Java interpreter before any objects are made.
- The keyword **void** simply tells the compiler that main( ) does not return a value.
- **String args[ ]** declares a parameter named **args**, which is an array of instances of the class **String**
  - Objects of type **String** store character strings
  - In this case, **args** receives any command-line arguments present when the program is executed.

```
class Dog{
String breed;
String color;
void  bark()
    {
    System.out.println("The " +color+" " +breed+ " is
    barking");
    }
public static void main(String args[])
    {
     Dog d1=new Dog();
     d1.breed="Pug";
     d1.color="black";
     d1.bark();
    }
}
```

o/p:The black Pug is barking

System.out.println("Hello World");

- This line outputs the string "Hello World" followed by a new line on the screen.

- **System** is a predefined class that provides access to the system
- **out** is the output stream that is connected to the console
- **println( )** displays the string which is passed to it

Another Example:

```
class Example2
{
 public static void main(String args[])
{
    int num;        // this declares a variable called num
    num = 100;   // this assigns num the value 100
    System.out.println("This is num: " + num);
    num = num * 2;
    System.out.println("The value of num * 2 is " + num);
  }
}
```

o/p : This is num: 100

The value of num*2 is 200

Note : - **Set Environment variables – To run java programs anywhere**

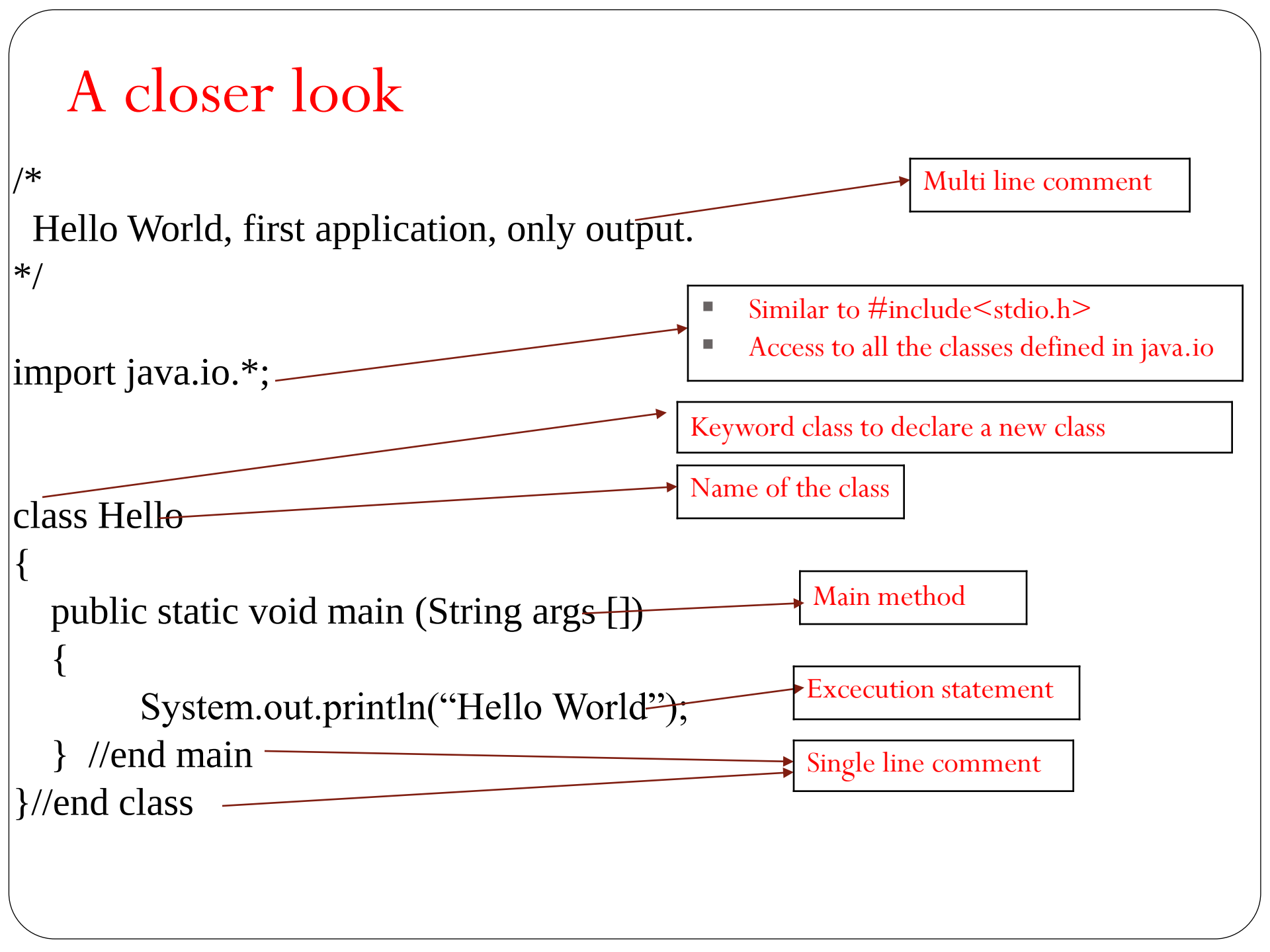
- Set path to JDK bin directory
  - set path=C:\Program Files\Java\jdk1.8.0_31\bin

A closer look

```
/*
  Hello World, first application, only output.
*/

import java.io.*;

class Hello
{
   public static void main (String args [])
   {
         System.out.println("Hello World");
   }  //end main
}//end class
```



Basic Structure of Java Program

A Java program involves the following sections:

- Documentation Section

  You can write a comment in this section. Comments are beneficial for the programmer because they help them understand the code.

- Package Statement

  A package is a group of classes that are defined by a name. That is, if you want to declare many classes within one element, then you can declare it within a package. It is an optional part of the program.There are built in as well as user defined packages

eg) package student;

- Import Statements

  This line indicates that if you want to use a class of another package, then you can do this by importing it directly into your program.

  eg) import student.MyClass;

Contd..

- Interface Statement

  Interfaces are like a class that includes a group of method declarations. It's an optional section and can be used when programmers want to implement multiple inheritances within a program.

- Class Definition

  A Java program may contain several class definitions. Classes are the main and essential elements of any Java program.

- Main Method Class

  Every Java stand-alone program requires the main method as the starting pointof the program. This is an essential part of a Java program. There may be many classes in a Java program, and only one class defines the main method. Methods contain data type declaration and executable statements.

Notice:

- Java is CASE SENSITIVE!!
- File name has to be the same as class name in file.
- Need to import necessary class definitions
- All statements in Java end with a semicolon.
- Whitespace is ignored by compiler
- In Java, all code must reside inside a class.