Object-Oriented Programming

Chapter One: Introduction to Object-Oriented Programming (OOP)

Agenda

- Overview of OOP
- Objects
- Classes
- Constructors
- Creating objects
- Accessing Objects
- Class declaration
  - Data Fields (instance or static variables)
  - Methods (instance or static methods)
- Modifiers
- this keyword
- Set and get functions

Overview of OOP

- A program is a set of instructions that tells a computer what to do in order to come up with a solution to a particular problem.
- There are a number of alternative approaches to the programming process, referred to as programming paradigms.
- Different paradigms represent fundamentally different approaches to building solutions to specific types of problems using programming.
- Most programming languages fall under **one paradigm**.
- Two of the most important programming paradigms are the **procedural paradigm** and the object-oriented paradigm.

Overview of OOP Cont.

- **Procedural programming** uses a list of instructions to tell the computer what to do step-by-step.
  - A procedure contains a series of **computational steps** to be carried out.
  - Procedural programming is also referred to as imperative programming.
  - Procedural programming languages are also known as top-down languages .
  - Examples of procedural languages include Fortran, COBOL and C, which have been around since the 1960s and 70s.

Overview of OOP Cont.

- **OOP is an approach** to problem-solving where all computations are carried out using objects.
- An object is a component of a program that knows how to perform certain actions and how to interact with other elements of the program.
- A method in object-oriented programming is like a procedure in procedural programming.
- The key difference here is that the method is part of an object.
- In object-oriented programming, you organize your code by creating objects, and then you can give those objects properties and you can make them do certain things.
- A key aspect of object-oriented programming is the use of classes.
- A class is a blueprint of an object. Example of OOP language Java, C#, C++, Visual Basic etc…

Overview of OOP Cont.

- Unlike other examples of computer programming language, object oriented programming focuses on the use of objects instead of actions in order to carry out tasks.
- That is the focus of oop is to break down a programming task into objects with each "object" encapsulating its own data and methods (subroutines).
  - Contrast it with procedural programming which break down a programming task into a collection of variables, data structures, and subroutines.
- Why OOP?
  - OO enhances key software quality factors of a system and its constituent components

Java

- Java
  - was created in 1991 and formally announced in 1995
  - by James Gosling et al. of Sun Microsystems.
  - Initially called Oak, latter was changed to Java because there was already a language called Oak.

Early History Website:
http://www.java.com/en/javahistory/timeline.jsp

Java Cont.

- The original motivation for Java
  - The need for platform independent language
- One of the first projects developed using Java
  - a personal hand-held remote control named Star 7.
- At about the same time, the **World Wide Web** and the Internet were gaining popularity.
- Gosling et. al. realized that Java could be used for Internet programming.

Java Cont.

- Java technology is more than just a programming language.
- It is a completely new approach to software development.
- Java Technology mainly consists two parts : **JDK** and JRE.

Java Development Kit (JDK)

- JDK is A free software development package from Sun Microsystems that implements the basic set of tools needed to write, test and debug Java applications and applets.
- It is the **toolkit** for developers that includes the Java compiler and the runtime environment.
- To write Java programs, you need the JDK installed.
- The major component of JDK are
  - **Java compiler** (.java file -> .class file)
  - **Java launcher** and **jdb debugger**

Java Development Kit (JDK) Cont.

- JDK Editions

Java Standard Edition (J2SE)

- J2SE can be used to develop client-side standalone *applications* or *applets*.

Java Enterprise Edition (J2EE)

- J2EE can be used to develop server-side applications such as *Java servlets* and Java *ServerPages*.

Java Micro Edition (J2ME)

- J2ME can be used to develop applications for *mobile devices* such as *cell phones*.

Java Runtime Environment (JRE)

- **JRE** is the program that emulates the **JVM**, so that users can run Java programs.
- To run Java programs, you need download and install the JRE.
- Runs code compiled for a JVM and performs class loading (through the class loader), code verification (through the bytecode verifier) and finally code execution.
- The major component of JRE are
  - Java platform core classes libraries
  - Java virtual machine (JVM)

Characteristics of Java

- **Java Is Simple**
  - Java is partially modeled on C++, but greatly simplified and improved.
  - Some people refer to Java as "C++--" because it is like C++ but with more functionality and fewer negative aspects.
- **Java Is Distributed**
  - Distributed computing involves several computers working together on a network.
  - Java is designed to make distributed computing easy.
  - Since networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file.

Characteristics of Java Cont.

- **Java Is Object-Oriented**
  - Java is inherently object-oriented.
  - Although many object-oriented languages began strictly as procedural languages, Java was designed from the start to be object-oriented.
- Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.
  - One of the central issues in software development is how to reuse code.
  - Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism.

Characteristics of Java Cont.

- **Java Is Interpreted**
  - You need an interpreter to run Java programs.
  - The programs are compiled into the **Java Virtual Machine** code called **bytecode**.
  - The **bytecode** is machine-independent and can run on any machine that has a Java interpreter, which is part of the **JVM.**
- **Java Is Multithreaded**
  - Multithread programming is smoothly integrated in Java, whereas in other languages you have to call procedures specific to the operating system to enable multithreading.

Characteristics of Java Cont.

- **Java Is Robust**
  - Java compilers can detect many problems that would first show up at execution time in other languages.
    - Java has eliminated certain types of error-prone programming constructs found in other languages.
  - Java has a runtime exception-handling feature to provide programming support for robustness.
- **Java Is Secure**
  - Java implements several security mechanisms to protect your system against harm caused by stray programs.

Characteristics of Java Cont.

- **Java Is Architecture-Neutral**
  - Write once, run anywhere
  - With a JVM, you can write one program that will run on any platform.
- **Java Is Portable**
  - Because Java is architecture neutral, Java programs are portable.
  - They can be run on any platform without being recompiled.

Characteristics of Java Cont.

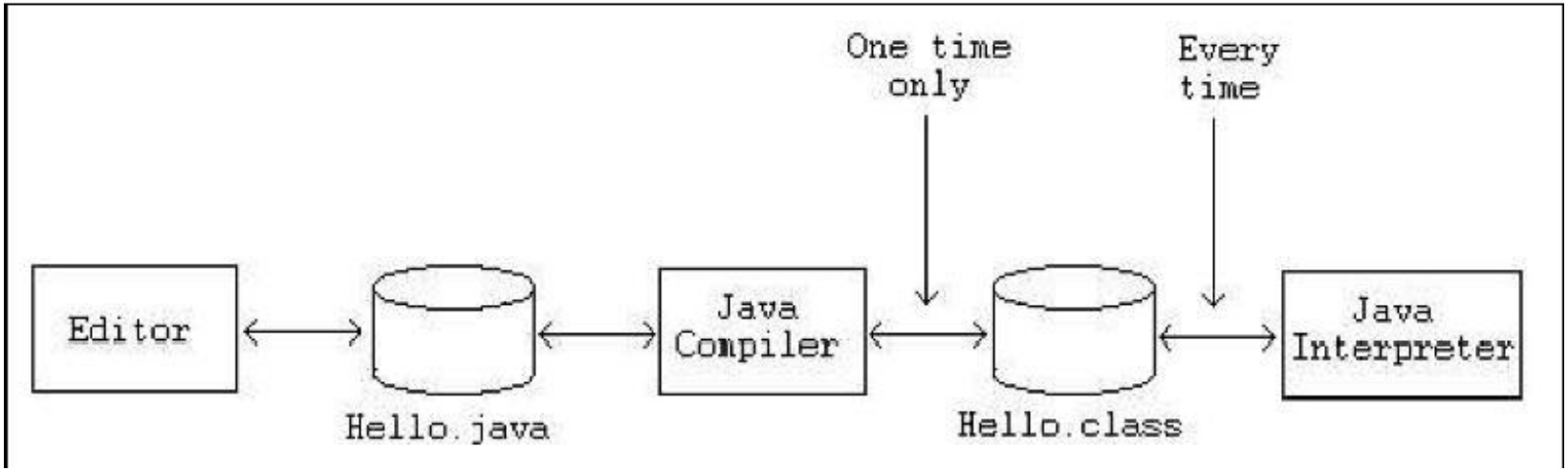
- **Java Is Dynamic**
  - Java was designed to adapt to an evolving environment.
  - New code can be loaded on the fly without recompilation.
  - There is no need for developers to create, and for users to install, major new software versions.
  - New features can be incorporated transparently as needed.
- **Java's Performance**
  - It has been said that java is 20 times slower than C/C++.
  - In other words, Java is too heavy to load – this is a big disadvantage of Java.

Why Java?

- The answer is that Java enables users to develop and deploy applications on the
  - desktop computers, and ….Stand Application
  - Internet for servers …..Servlets, Applets
  - small hand-held devices …..g. Mobile Computing
- The future of computing is being profoundly influenced by the Internet, and Java promises to remain a big part of that future.
  - Java is a general purpose programming language.
  - Java is the Internet programming language.

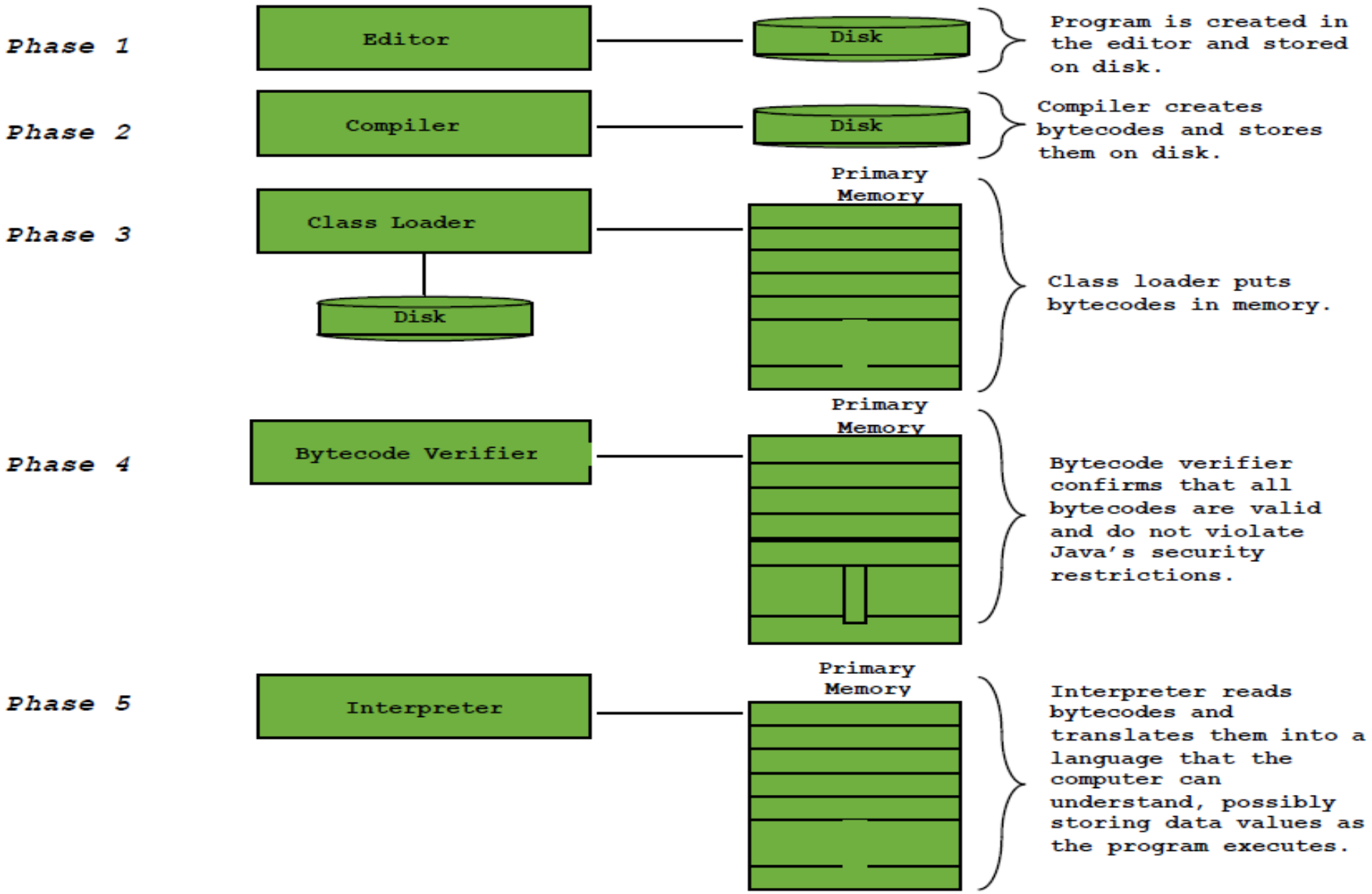
Phases of a Java Program

- Java programs normally go through five phases – *edit*, *compile*, *load*, *verify* and *execute*.



Task	Tool to use	Output
Write the program	Any text editor	File with .java extension
Compile the program	Java Compiler	File with .class extension (Java bytecodes)
Run the program	Java Interpreter	Program Output

Phases of a Java Program Cont.



Phase 1: Creating a program

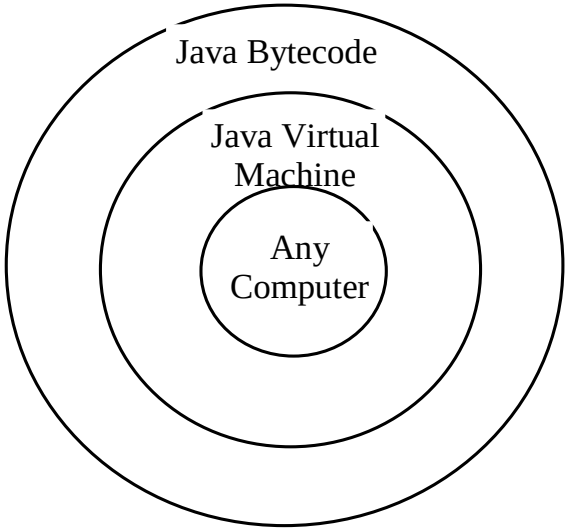
- Involves editing a file with an editor
- The following are some Java IDEs
  - Jbuilder, NetBeans , Sun ONE Studio, Eclipse, jEdit, Jcreator, BlueJ, jGRASP, etc
  - Most of this editors can be downloaded for free
  - Any text editor can also be used for creating Java programs
  - Programs written on IDEs or any other editors are known as **Source Code**.
  - Java source code file names end with **.java** extension

Phase 2: Compiling Java Programs into Bytecodes

- Since a computer cannot understand a source program, program called a **compiler** is used to translate the source program into a machine language program called an **object program**.
- Java was designed to run object programs on any platform.
- With Java, you write the program once, and compile the source program into a special type of object code, known as **bytecode**.
- The **bytecode** can then run on any computer with a Java Virtual Machine

Phase 2: Compiling Java Programs into Bytecodes Cont.

- A **virtual machine** is a software application that simulates a computer-JVM is one of the most widely used virtual machine
- Java Virtual Machine, which is part of JRE, is a software that interprets Java bytecode.
- To compile Java Program
  - **Javac welcome.java**
- To execute java programs
  - **Java welcome**



Phase 3: Loading a program

- A program must be placed in memory before it can execute a process known as **loading**.
- The class loader takes the .class file (produced during compilation process) containing the program's bytecodes and transform them to primary memory.
- The class loader also loads any of the .class files provided by java that your program uses.
- The .class files can be loaded fro a disk on your system or over a network

Phase4 : ByteCode Verification

- involves examining bytecodes to ensure that they are valid and do not violate **Java's security restriction**.
- Java enforces strong security, to make sure that Java programs arriving over the network do not damage your files or your system (as computer viruses and worms might)

Phase 5: Execution

- The JVM executes the program's bytecodes, thus performing the actions specified by the program.
- JVMs typically execute bytecodes using a combination of **interpretation** and so-called **just-in-time (JIT)** compilation.
  - In this process, The JVM analyzes the bytecodes as they are interpreted, searching for hot spots— parts of the bytecodes that execute frequently.
- In this phase bytecodes are translated into the underlying computer's machine language.

A Simple Java Program

- The following code is a java program that prints the string " *Welcome to Java Programming!*".

```
1.  /*  Your first Java program; Welcome.java
2.      The program prints a string Welcome to Java Programming
3.  */
4.  public class Welcome {

5.      //main method begins execution of Java application
6.      public static void main (String args[] ){
7.          System.out.println( "Welcome to Java Programming!" );
8.      }//end  of main method
9.  } //end of class Welcome
```

A Simple Java Program

- Lines 1-3 & 5 comment lines
  - There are two types of comments in Java
    - End-of-line (Single line) comments
      - A line that begins with //
      - A // comment also can begin in the middle of a line and continue until the end of that line
    - Traditional (Multiple line) Comments
      - A text that begins in /* ends in */
      - All text between these delimiters is comment so it is ignored by the compiler.
      - These comment can be applied for a line, more than a lines or part(s) of a line

A Simple Java Program Cont.

- Javadoc comments
  - A text that begins in /** and ends in*/
  - As with traditional comments, all text between the Javadoc comment delimiters is ignored by the compiler.
  - Javadoc comments enable programmers to embed program documentation directly in their programs.
- Line 4: *public class Welcome {*
  - This line begins class declaration of the class Welcome
  - Every program in Java consists of at least one class declaration .
  - *Generally, Java class decleration has the following format*

    [Access level Specifier] class class_Name { … }

A Simple Java Program Cont.

- Access level Specifier
  - could be omitted (none), **public**, **private**, or **protected**
  - These are keywords that help set the *visibility* and *accessibility* of a class, its member variables, and methods.
  - They determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class.
- class
  - The class keyword introduces a class declaration in Java and is immediately followed by the class name (Welcome1).
- class_Name
  - Is the name of a class. A Java class name is an identifier.
  - By convention class names should start with capital letters
  - Java is case Sensitive

A Simple Java Program Cont.

- *Line 6: public static void main (String args[]){*
  - Is a method of which the starting point of every java application.
  - Every Java application must contain one main method
  - public is a keyword which specifies access level of the method.
  - Static is another keyword [its meaning is to be discussed latter]
  - void is a return type that indicates that this method performs a task but will not return any information when it completes its task.
  - String args[] … is an array of strings which is an argument of the main function.

A Simple Java Program Cont.

- Line 7:
  - System.out.println("Welcome to Java Programming!");
    - This line prints the string contained in quotation
    - **System.out** is known as the standard output object.
    - When **System.out.println** completes its task, it positions the output cursor (the location where the next character will be displayed) to the beginning of the next line while **system.out.print** don't.
    - Every statement of java must end in semicolon (;) so is line 7
- Lines 8 and 9
  - This line depicts the end of the main method and the class welcome.

Object

- An *object* represents an entity in the real world that can be distinctly identified.
  - For example, a student, a desk, a circle, a button, etc.
- An Object is a self-contained element of a computer program that represents a related group of features and is designed to accomplish specific tasks.
- Each object has a specific role in a program, and all objects can work with other objects in defined ways.
- Languages that follow object-oriented concepts describe the interaction among objects.

Object Cont.

- An object has both a state and behavior.
  - The state defines the object, and the behavior defines what the object does.
- The state of an object pertains to data elements and their associated values. Data elements associated with objects are called instance variables.
- The behavior of an object depends on the actions the object can perform on the instance variables defined within the object.
  - In procedural programming, such a construct would be called a function.
  - In object-oriented terminology, this construct is called a method.

Object, example

- If you create a software object that models your television.
  - The object would have variables describing the television's current state, such as
    - Its status is on,
    - the current channel setting is 8,
    - the current volume setting is 23, and
    - there is no input coming from the remote control.
  - The object would also have methods that describe the permissible actions, such as
    - turn the television on or off,
    - change the channel,
    - change the volume, and
    - accept input from the remote control.

Class

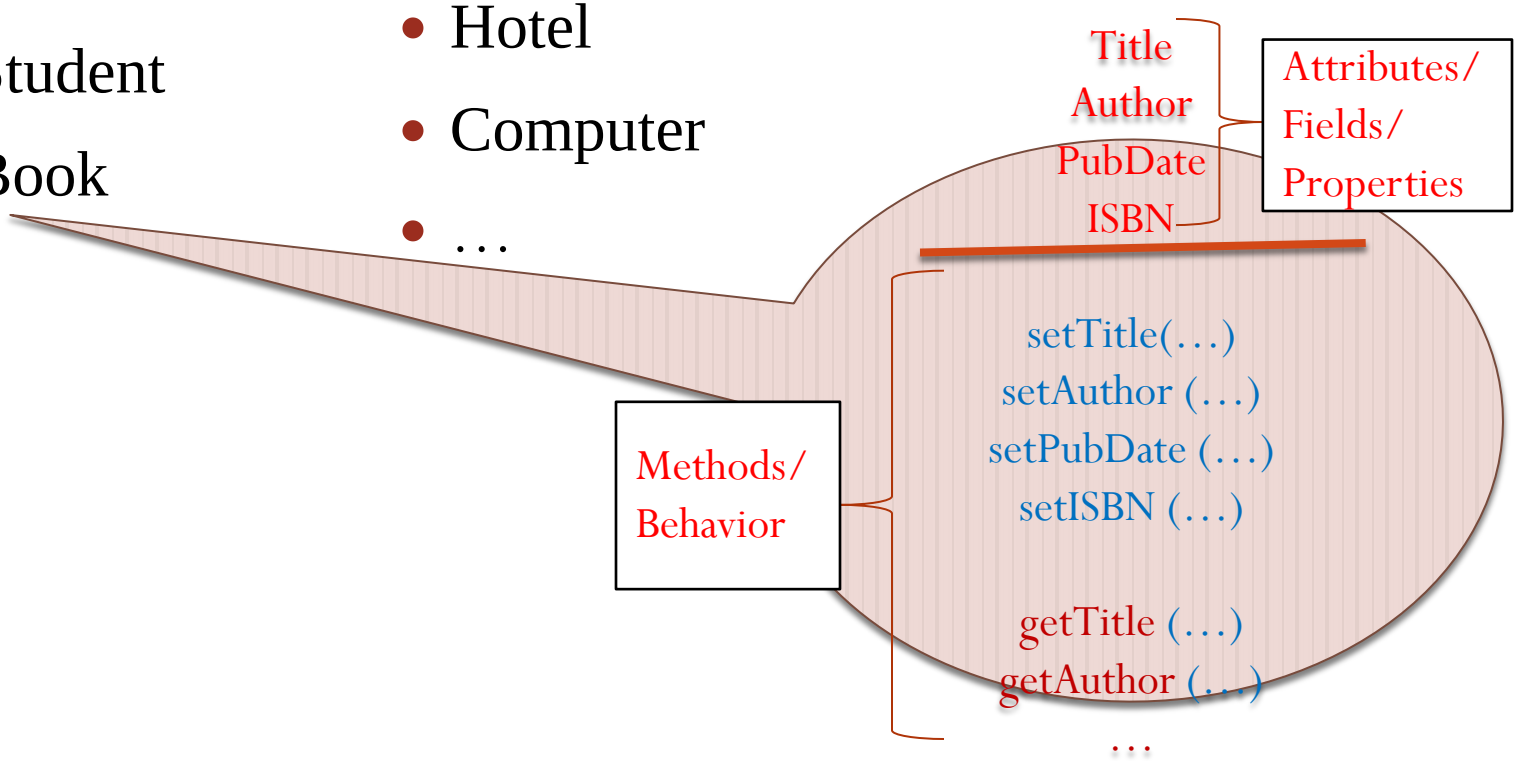
- In object-oriented programming, a **class** is a programming language construct that is used as a blueprint to create objects of that class.
  - This blueprint describes the state and behavior that the objects of the class **all share**.
  - An object of a given class is called an instance of the class.
  - The class that contains that instance can be considered as the type of that object, e.g. a type of an object of the "Fruit" class would be "Fruit".
    - That is a class essentially serves as a template for an object and behaves like a basic data type “int”.
  - A class usually represent a noun, such as a person, place or (possibly quite abstract) thing - it is a model of a concept within a computer program.

Class…

- A Java class uses *variables* to define *data fields* and *methods* to define *behaviors*.
- Additionally, a class provides a special type of methods, known as *constructors*, which are invoked to construct objects from the class.
- A single class can be used to instantiate multiple objects. This means that you can have many active objects or instances of a class.
- The object describing the functions of your television is an instance of a class of objects called television.
- Each object within a class retains its own states and behaviors.

Class…

- Concepts that can be represented by a class:
  - Tree
  - Man
  - Animal
  - Student
  - Book
  - Building
  - Car
  - Hotel
  - Computer
  - …



Classes, Example

```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

Data field

Constructors

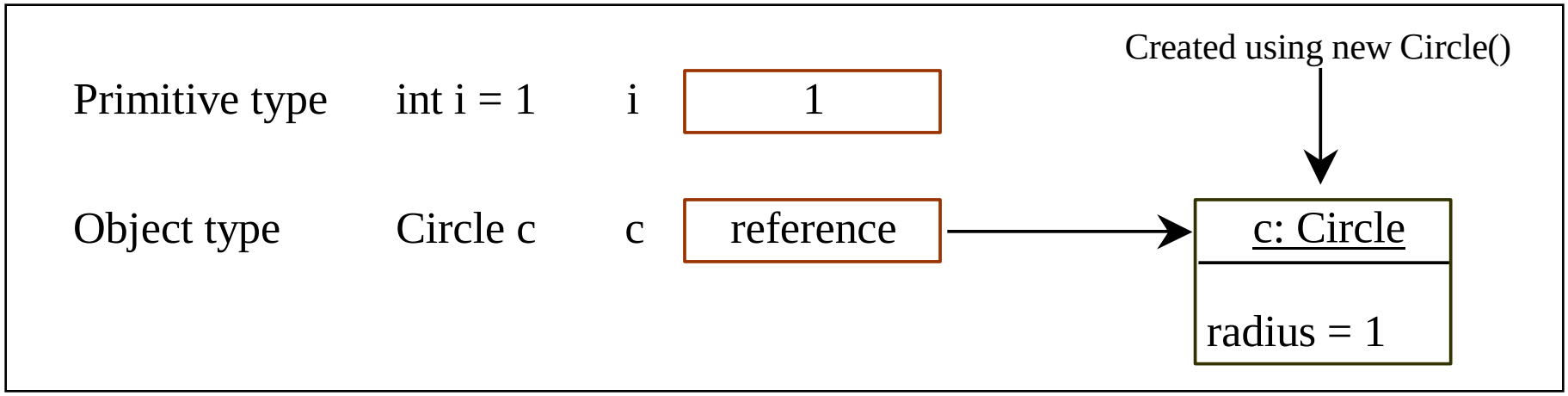
Overloading

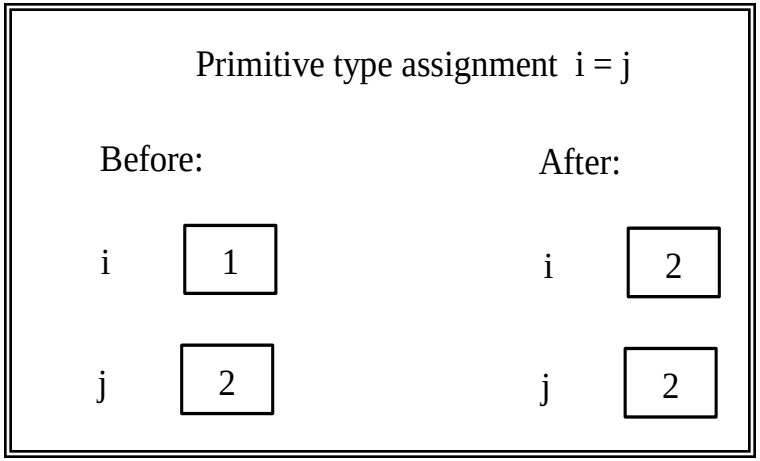
Method

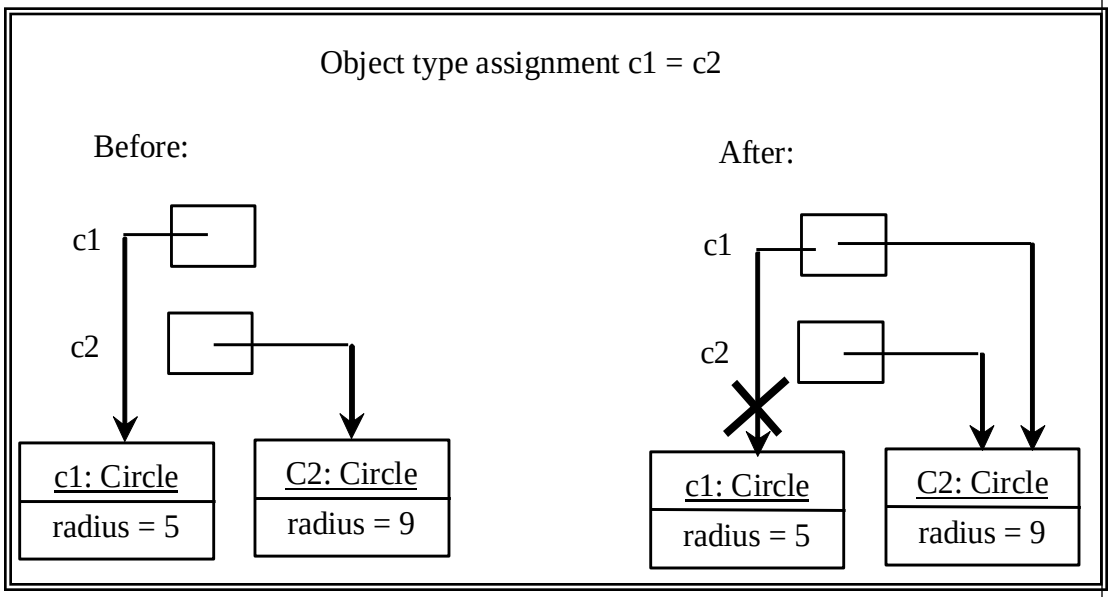
Creating Objects, Example

- Example
  - /*declare a reference variable and create an object using an empty constructor */
  - Circle c1 = new Circle();
  - /*declare a reference variable and create an object using constructor with an argument */
  - Circle c2 = new Circle(5.0)
  - //declare reference variables
  - Circle c3, c4 ;
  - /* create objects and refer the objects by the reference variables
  - c3 = new Circle();
  - c4 = new Circle(8.0);

Primitive Data Types and Object Types







Garbage Collection

- The object becomes a candidate for automatic garbage collection.
- Java automatically collects garbage periodically and releases the memory used to be used in the future.
  - That is the JVM will automatically collect the space if the object is not referenced by any variable .
- On the figure on previous slide object c1 is no more referenced and cannot be used in future

Accessing Objects

- Once objects have been created , its data fields and methods
  - Referencing the object's data:

    objectRefVar.data

    e.g., myCircle.radius = 100;

  - Invoking the object's method:

    objectRefVar.methodName(arguments)

    e.g., double area = myCircle.getArea();
- This is why it is called "object-oriented" programming; the object is the focus.

Classes declaration

- Class declaration has the following syntax

  ClassModifiers(optional) **class** ClassName pedigree(optional){

  //Class body;

  }

- Class Body may contain
  - data fields
    - [FieldModifier(s)] returnType varName ; //declaration
  - constructors
    - [Modifiers] ClassName(…){…}
  - Member functions
    - [MethodModifier(s)] returnType methodName(…){…} //declaration
  - and even other classes

- Pedigree - extends, implements

Data fields

- Data Fields are variables declared directly inside a class (not inside method or constructor of a class)
- Data fields can be variables of primitive types or reference types.
- Example

```
public class Student {
   String name;
   int age;
   char gender;
}
```

- The default value of a data field is null for a reference type, 0 for a numeric type and false for a boolean type. However, Java assigns no default value to a local variable inside a method.

Data fields…

```
public class Test {
  public static void main(String[] args) {
    Student student = new Student();
    System.out.println("name? " + student.name);
    System.out.println("age? " + student.age);
    System.out.println(" isSci? " + student.isSci);
    System.out.println("gender? " + student.gender);
  }
}
```

- Data fields can be initialized during their declaration.
- However, the most common way to initialize data fields is inside constructors

Member Functions

- A function declared inside the class definition is a member function
  - Exception: "static member functions" are not really member functions, more on that later
- Methods implement the behavior of classes
  - That is method is an operation which can modify an objects behavior.
  - In other words, it is something that will change an object by manipulating its variables.

- **Only an object's methods should modify its variables**

Instance Variables and Methods

- Instance variables and methods (also called members) are those associated with individual instances (or objects) of a class
- Instance variables belong to a specific instance.
- Instance methods are invoked by an instance of the class.
- To get to the value of an instance variable or to invoke instance methods, you use an expression in what's called dot notation.
- Example
  - Circle cl = new Circle ();
  - cl.radius = 5;
  - System.out.println("Area " + cl.getArea());

class member visibility

- Java provides several modifiers that control access to data fields, methods, and classes. This section introduces the **public**, **private**, and **default modifiers**.
- **public** makes classes, methods, and data fields accessible from any class.
- private makes methods and data fields accessible only from within its own class.
- **If public or private is not used**, then by default the classes, methods, and data fields are accessible by any class in the same package.
  - This is known as *package-private or package-access*.

Chapter One End

Thank You!!!