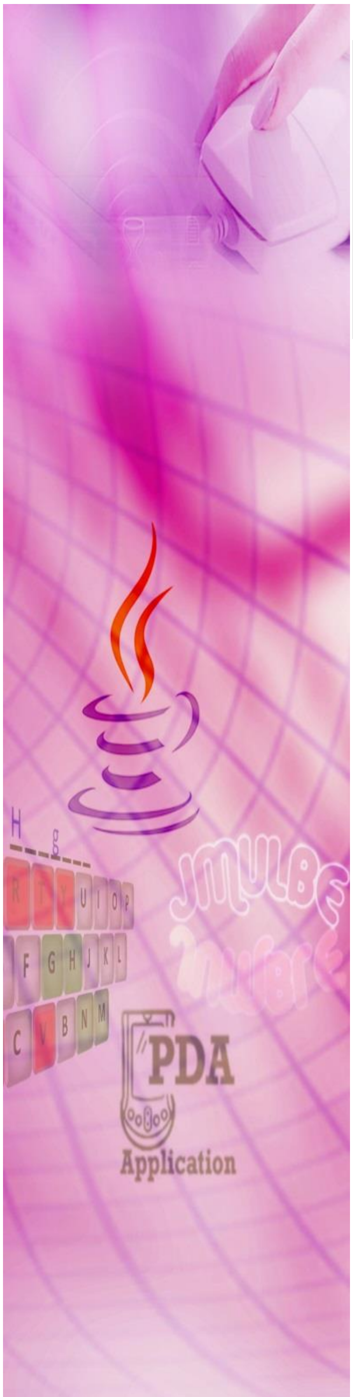

Instructor Inputs

Notes for Faculty

The faculty needs to consider the following points for this course:

- The study material for this course is divided into three books, Introduction to Java – Student Guide (SG), Activity Book, and Toolbook. The SG contains the concepts, and the Activity Book provides the activities to help the learners understand how to implement these concepts. The Toolbook provides instructions to install and use the software required for this course. The SG contains the placeholders for the activities.
- While going through the SG, whenever you find the placeholder for an activity or a task, you need to refer to the Activity Book to see the steps for performing the activity.
- For conducting the classroom sessions, you need to refer to the SG, Toolbook, and the Activity Book.
- For conducting the machine room session, you need to refer to the Activity Book that lists the exercises to be performed in these sessions.
- While developing the activities, exercises, and the additional exercises, the implementation of concepts taught in the SG are kept in focus. If students want, they can add the code to implement validations of inputs.
- The presentation slide contains animations that require .swf media files. Make sure that these .swf media files and .ppsx presentation files are at the same location.

Session Overview

This session covers Chapter 1 of the book, Introduction to Java – SG. In this session, the students will learn the basic concepts of the Java programming language. They will learn the features of Java, the various components of the Java architecture, and the building blocks of a Java program. In addition, they will learn how to access the various class members in a Java program.

Introducing Java

Handling Tips

Start the session by asking the students if they know about a programming language. Elicit the responses and briefly discuss the concepts of object-oriented programming and a programming language. Then, start explaining the importance of Java as a programming language and explain each of its features along with the information given in the Additional Inputs topic. Next, tell them that in order to create and execute Java programs, it is essential to have a basic understanding of the architecture of Java. Explain the various components of the Java architecture with the help of the diagram given in the SG.

Additional Inputs

Garbage collection is a mechanism by using which a programmer need not explicitly free the allocated memory. Each object within a program uses some amount of the system resources, such as the system memory, during run time. Garbage collection is a process that is used to free the memory of the objects that are no longer in use, so that it can be reallocated. In this way, garbage collection serves as a mode for memory recycling where the space occupied by one object is recycled so that it is made available for the subsequent new objects.

The garbage collection process is run automatically by the JVM. However, if you have a process that requires a large amount of memory, then you can inform the JVM that you want the garbage collector to be run. This can be done by invoking the `System.gc()` method in your program code. However, invoking the preceding method will not ensure that garbage collection will be performed.

In its lifetime, an object may acquire several resources, such as file. At the end of its life, an object must release all its acquired resources. For this, Java supports the concept of finalization. The process of finalization involves invoking the `finalize()` method of an object before reclaiming its memory. The `finalize()` method can contain the code to release the resources of the object.

Identifying the Building Blocks of a Java Program

Handling Tips

Tell the students about the basic building blocks of a Java program. Explain the importance of each of the building blocks and relate it to the development of the Classic Jumble Word application. To ensure a better understanding, explain the code snippets to the students. Explain the concept of classes to the students in the context of the Classic Jumble application. Also, tell the students about the naming conventions for the classes and files. While explaining the class members, emphasize on literals and their types and constructors and objects.

Additional Examples

Identifying Class Members

The following code demonstrates the usage of a method-local inner class:

```
public class InnerDemo {
    private int speed = 500;
    void display()
    {
        final int retardation = 50;
        class LocalDemo
        {
            void show()
            {
                System.out.println("The speed is " + speed);
                System.out.println("The retardation is " + retardation);
            }
        }
        LocalDemo l = new LocalDemo();
        l.show();
    }
    public static void main(String[] args) {
        InnerDemo obj = new InnerDemo();
        obj.display();
    }
}
```

In the preceding code, the class, `LocalDemo`, is a method-local inner class, as it is created inside the `display()` method of the outerclass, `InnerDemo`. The `LocalDemo` class implements the `show()` method. It prints the value of the `speed` instance variable of the `InnerDemo` class and the local variable `retardation` of the `display()` method.

The following rules apply to the method-local inner classes:

- They cannot be accessed outside the method in which they are created.
- The local variable defined inside the method that contains the method local inner class can be accessed from the method-local inner class only if it is declared final. Otherwise, the program will not compile.

Accessing Class Members

Handling Tips

Tell the students that class members describe the characteristics and behavior of an object and need to be accessed to implement the desired functionality. Explain how the class data members can be accessed by the objects, along with the syntax and a relevant example.

Next, tell them that while creating enterprise level applications or real time applications, they might need to hide or protect certain class members from other classes in a Java application. Tell them that restricting access to the members of a class is a way of achieving information hiding or encapsulation. Then, explain the various access specifiers in Java. Tell the students that the `private` access specifier is the most restrictive access specifier, whereas the `public` access specifier is the least restrictive specifier. For a better understanding, illustrate the concept of access specifiers diagrammatically. In addition, demonstrate the use of access specifiers with help of the code given in the Additional Examples topic.

Next, tell them that to determine or define how data members and methods are used in other classes and an object, Java provides various access modifiers. Explain each one of the various modifiers and provide relevant examples. Emphasize on the usage of the `static`, `final`, and `abstract` keywords as they are used more commonly with respect to the other modifiers.

Additional Examples

Using Access Specifiers

The following code demonstrates the usage of access specifiers:

```
public class Demo
{
    private int account_no = 1432; /* Data members converted to private to
encapsulate data */
    private String name = "Shawn";
    private int age = 64;
    protected int bankID = 113;
    private float balance = 10000;
    public void show() // Method can be called from outside the class to
access the data members
    {
        System.out.println("Name of this customer is= " + name);
        System.out.println("Age =" + age);
        System.out.println("Balance of this customer is= " + balance);
    }
}
class Demo2 extends Demo
{
    void showID()
    {
        show();
        System.out.println("Bank ID of this customer is= " + bankID);
        System.out.println("Bank ID of this customer is= " + Demo.counter);
    }
    public static void main(String args[]) {
        Demo2 d1 = new Demo2();
        d1.showID();
    }
}
```

The output of the preceding code is:

```
Name of this customer is= Shawn
Age =64
Balance of this customer is= 10000.0
Bank ID of this customer is= 113
```

Activity 1.1: Creating and Executing a Class

Handling Tips

Discuss the problem statement with the students.

The solution file, **Hangman.java**, for this activity is provided at the following location in the TIRM CD:

- **Datafiles For Faculty\Activities\Chapter 01\Activity 1.1\Solution**

FAQs

- *Do we need to create an object for using static variables or methods?*

  Ans: You do not need to create an object of a class to access its static variables or methods. You can use these variables or methods with their class name.

- *Is it possible to convert one data type into another?*

  Ans: Yes, it is possible convert one data type into another by using the type casting and autoboxing techniques in Java.

- *How can we write comments in a Java program?*

  Ans: In order to write a single line comment, you can precede it with `//`. If you want to write multiple line comments, you need to enclose them in `/*` and `*/`, as shown in the following code snippet:

```
int a; // single line comment
/*
       System.out.println("Name of this customer is= "+ name);
       System.out.println("Age ="+ age);
       System.out.println("Balance of this customer is= "+ balance);
*/
```