BASICS OF OBJECT ORIENTED PROGRAMMING

Learning Objectives

After studying this lesson the students will be able to:

- Understand the need of object oriented programming
- Define the various terms related to object oriented programming
- Identify the features of an object oriented programming language
- Use features of object oriented programming language to develop simple applications

Over the last lesson, we have reviewed the core Java language. We have learnt how to work with variables and data, perform operations on that data, make decisions based on the data and also loop repeatedly over the same section of code. Now we will move on to learn a concept that is central to Java, namely Object Oriented Programming. Object Oriented Programming is a very user friendly yet a powerful approach to solve basic to difficult problems. The idea was created for developing a language that could be used for system description (for people) and system prescription (as a computer program through a compiler). There are several object-oriented languages. The three most common ones are, Smalltalk, C++ and Java.

Puzzle [4]

A student and a lady are travelling in a train. They get around talking and the lady decides to give a puzzle to the student. She tells him that she has 3 children whose product of ages is equal to the maximum number of runs possible to score in an over without any illegitimate ball being bowled (i.e. NB, Wide, etc). Also, the sum of their ages is equal to her berth number. However, the student isn't able to answer. The lady then gives him a further hint that the eldest of her children has only one eye. At this information, the

student knows the ages of the three children. Without knowing the lady's berth number, can you guess the ages of her children?

Introduction

James Alexander is a resident of a developed nation and works as a freelance consultant. He is hired by one of the corporate houses of a developing nation to plan a strategy to improve production in one of their factories which is located in a remote village named Khabri. The consultant decides to submit a quick action plan and so starts searching for information about the remote village. He has never visited any of the remote locations and so tries to simply imagine the problems faced by remote people. Mohan Swamy is a resident of one of the developing countries and he also is a freelance consultant. He completed his studies from a top notch university and to actually put his theoretical knowledge to practice, he started staying in the remote village Khabri. He wanted to actually experience the hardships faced by people residing in remote areas. To sustain himself he decides to pick up a job in the only factory situated in Khabri. The HR manager impressed with his in-depth knowledge and qualifications requests him to also plan a strategy to improve production of their factory. Who do you think will be able to provide a more viable solution? The obvious answer for most of us would be that the person sitting in the remote village and literate enough to solve the problem will be able to provide a better strategy because he closely understands the real problems of the residents as compared to a person sitting far away. But what does this teach us about programming? This teaches us that in programming also functions/methods/programs written for specific situations are able to manipulate data of their respective entities more efficiently. Now, let us understand a little about the various programming paradigms.

Introduction to Programming

Computer programming is a process of designing, writing, testing, debugging and maintaining the source code of computer programs written in a particular programming language. The purpose of programming is to organize instructions that are capable of solving real life problems. The process of writing source code of programs requires expertise in subject, knowledge of desired application domain, a formal logic and knowledge of syntax of the relevant programming language. Writing instructions in the desired order gives the required results from the program but when these instructions

increase in number, it becomes extremely difficult for the programmer to debug, maintain or troubleshoot codes. For this reason, technology experts kept developing and introducing different programming paradigms and accordingly kept developing languages to support these paradigms. Procedural programming paradigm was one of the major stepping stone for these experts which focused on breaking down a programming task into a collection of small modules known as sub routines or procedures. This paradigm helped the programmers to debug, maintain or troubleshoot codes in a more effective manner. The experts did not stop their research in improving this paradigm and introduced a new paradigm known as object oriented programming paradigm where a programming task was broken into objects. Here each object was capable of encapsulating its own data and methods (subroutines / procedures). The most important distinction is that where procedural programming uses procedures to operate on data, object oriented programming bundles data and methods together and operates as a independent entity of the program. Some languages support one particular programming paradigm while some are developed to support multiple programming paradigms. C++, C sharp, Object Pascal etc. are the languages which support procedural as well as object oriented paradigm. Java supports only object oriented programming. Basic, COBOL support only procedural programming.

Object Oriented Programming

Object Oriented Programming follows bottom up approach in program design and emphasizes on safety and security of data. It helps in wrapping up of data and methods together in a single unit which is known as data encapsulation. Object Oriented Programming allows some special features such as polymorphism and inheritance. Polymorphism allows the programmer to give a generic name to various methods or operators to minimize his memorizing of multiple names. Inheritance enables the programmer to effectively utilize already established characteristics of a class in new classes and applications.

The major components of Object Oriented Programming are as follows:

1. Class
2. Object

3. Data Members & Methods

4. Access Specifier and Visibility Modes

Classes and Objects :

A class is used to encapsulate data and methods together in a single unit. It helps the programmer to keep the data members in various visibility modes depending upon what kind of access needs to be provided in the remaining part of the application. These visibility modes are classified as private, public and protected. Usually, data members of a class are kept in private or protected visibility modes and methods are kept in the public visibility mode.

An object is an instance of a class that is capable of holding actual data in memory locations.

Class and objects are related to each other in the same way as data type and variables. For example, when we declare float variable named marks, the variable marks can be thought of as an object of type float which can be assumed as the class. If we take another hypothetical case in which Human is a class, Mr. Arun Shah, Mr. Aneek Ram will be the objects of this Human class.

Data Members and Methods :

We have already learnt that a class contains data members and methods. As discussed in the above example, Mr.Arun Shah is an object of class Human. The phone numbers retained by Mr.Arun Shah in his brain (memory) will be the data. His eyes, ears, nose and mouth can be considered as various methods which allow Mr.Arun Shah to collect, modify and delete data from his memory.

In real java programming, this data will be required to conform to a specific data type as in char, int, float or double whereas the methods will be a sequence of steps written together to perform a specific task on the data. Carefully observe the illustration given in Figure 4.1 to reinstate the theoretical concepts learnt above.

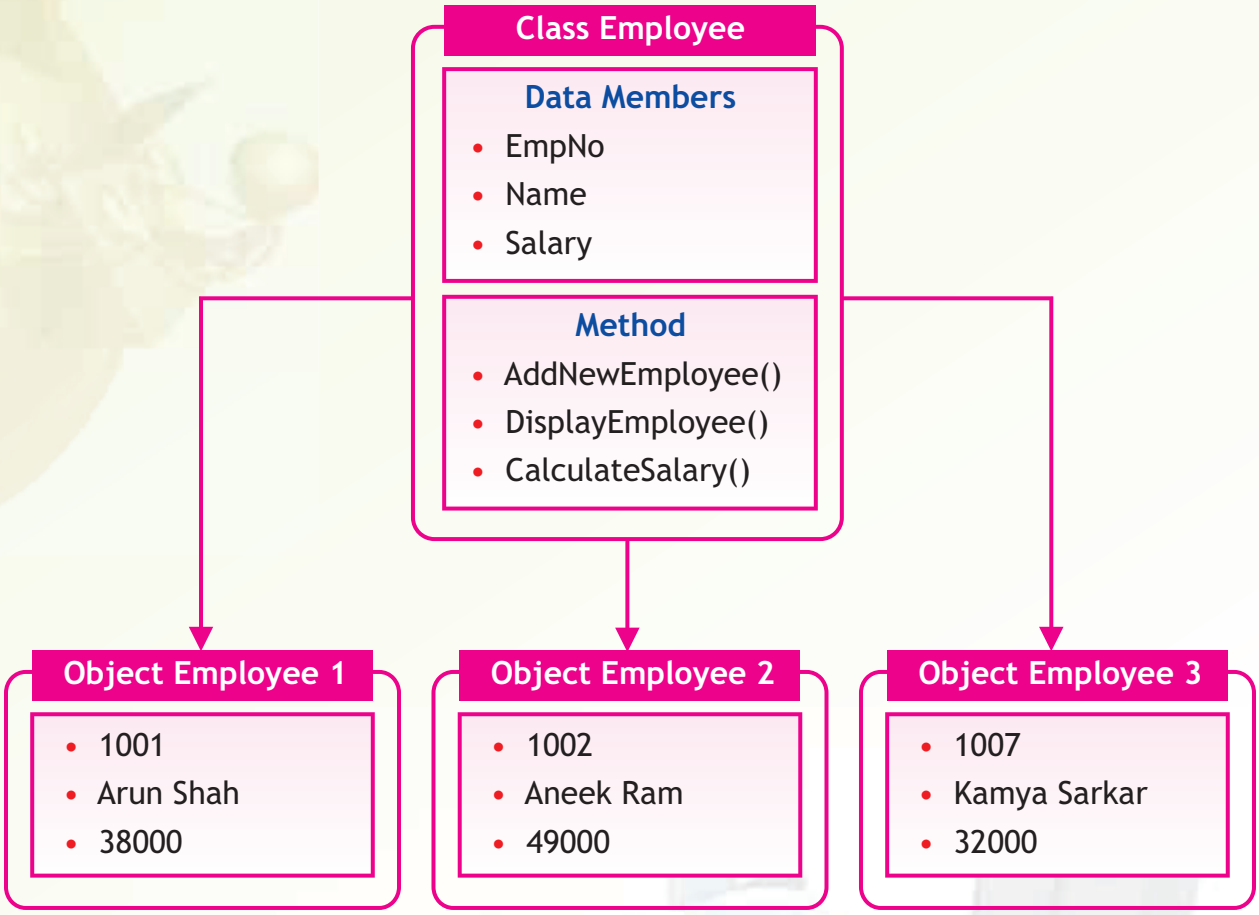

Figure 4.1 Illustration Showing the Class, Object, Members and Methods

Referring to the situation presented in the introduction of the chapter, just like the person residing in the village can efficiently solve problems pertaining to his village, similarly the methods of specific classes are able to manipulate data of their respective classes efficiently resulting in better security of data in an Object Oriented Programming paradigm.

Now that you are clear about the concept of a class and an object, you will be able to appreciate and identify classes and methods that we have already been using throughout our class XI. Do you know JTextField, JLabel, JTextArea, JButton, JCheckBox and JRadioButton are all classes and the jTextField1, jLabel1, jTextArea1, jButton1, jCheckBox1 and jRadioButton1 components are all objects. The setText(), setEnabled(), pow(), substring() are all methods of different classes. This concept is illustrated in Figure 4.2.

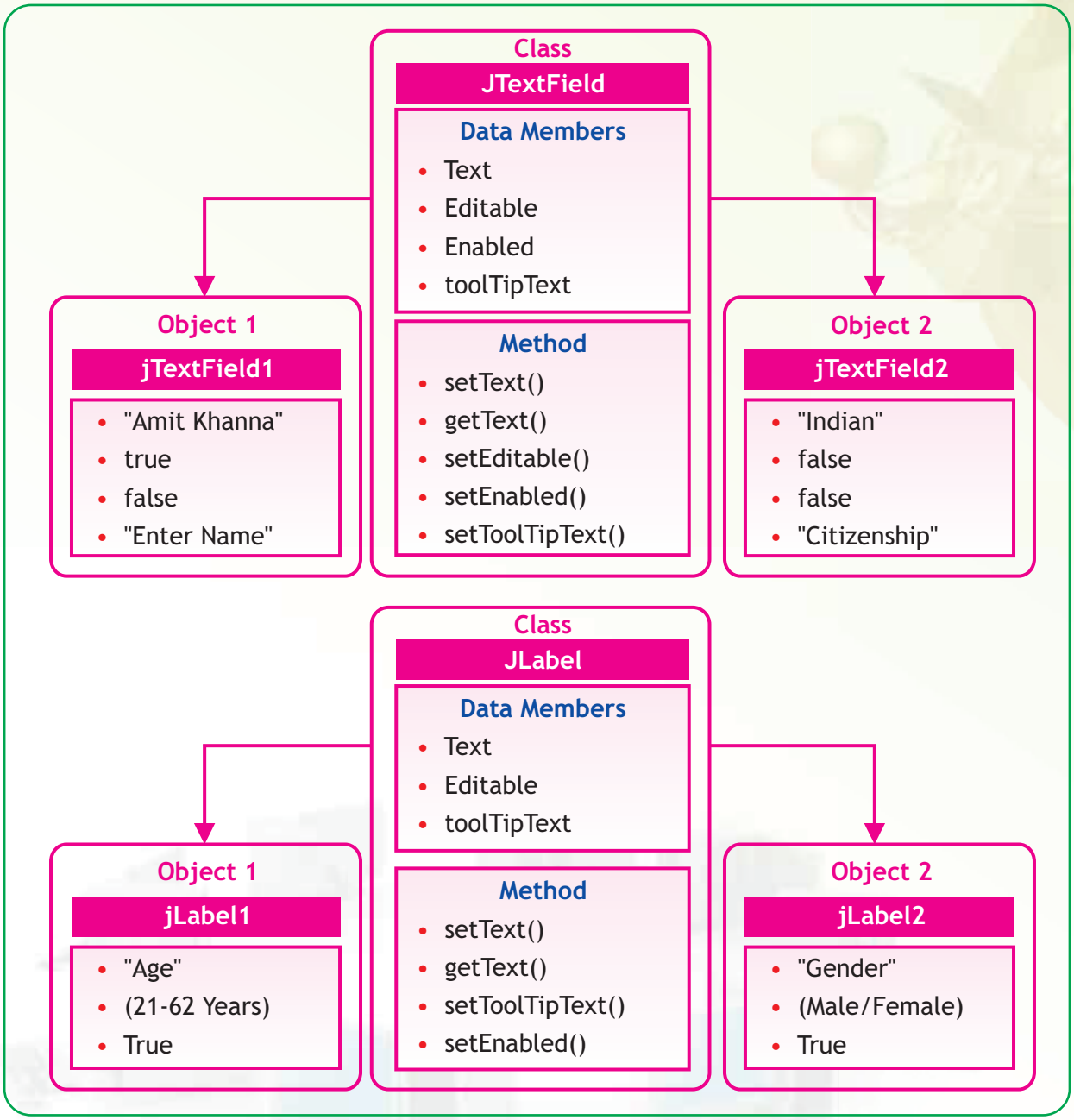

Figure 4.2 JTextField and JLabel Classes

Notice that the properties like Text, Enabled, Editable are actually the data members in the class because they store specific values as data. For example the property Text of jTextField1object contains the actual text to be displayed in the text field.

The following two tables summarize the data members (properties) and methods of all classes learnt during the course till now:

Class	Objects	Data Members (Properties)	Methods
JTextField	*jTextField1*	• *Text* • *Editable* • *Enabled* • *ToolTipText*	• *setText()* • *getText()* • *setEditable()* • *setEnabled()* • *setToolTipText()*
JLabel	*jLabel1*	• *Text* • *Enabled* • *ToolTipText*	• *setText()* • *getText()* • *setEnabled()* • *setToolTipText()*
JTextArea	*jTextArea1*	• *Columns* • *Editable* • *Font* • *lineWrap* • *Rows* • *wrapStyleWord* • *toolTipText*	• *isEditable()* • *isEnabled()* • *getText()* • *setText()*
JButton	*jButton1*	• *Background* • *Enabled* • *Font* • *Foreground* • *Text* • *Label*	• *getText()* • *setText()*

JCheckBox	*jCheckBox1*	• *Button Group* • *Font* • *Foreground* • *Label* • *Selected* • *Text*	• *getText()* • *setText()* • *isSelected()* • *setSelected()*
JRadioButton	*jRadioButton1*	• *Background* • *Button Group* • *Enabled* • *Font* • *Foreground* • *Label* • *Selected* • *Text*	• *getText()* • *setText()* • *isSelected()* • *setSelected()*
JPasswordField	*jPasswordField1*	• *Editable* • *Font* • *Foreground* • *Text* • *Columns* • *toolTipText*	• *setEnabled()* • *setText()* • *getText()* • *isEnabled()*
JComboBox	*jComboBox1*	• *Background* • *ButtonGroup* • *Editable* • *Enabled*	• *getSelectedItem()* • *getSelectedIndex()* • *setModel()*

		• *Font* • *Foreground* • *Model* • *SelectedIndex* • *SelectedItem* • *Text*	
JList	*jList1*	• *Background* • *Enabled* • *Font* • *Foreground* • *Model* • *SelectedIndex* • *SelectedItem* • *SelectionMode* • *Text*	• *getSelectedValue()*
JTable	*jTable1*	• *model*	• *addRow()* • *getModel()*

Polymorphism :

It is the ability of a method to execute in many forms. In object oriented programming there is a provision by which an operator or a method exhibits different characteristics depending upon different sets of input provided to it. This feature in Object Oriented Programming is known as polymorphism. Two examples of polymorphism are method overloading and operator overloading. Method overloading is where a method name can be associated with different set of arguments/parameters and method bodies in the same class. The round() method of the Math class and the substring() method of the String class are good examples of method overloading. The following examples explain how round() and substring() methods are overloaded.

If the argument passed to the round() method is of type double then it rounds off the double value and returns the closest long value. On the other hand, if the argument passed to the round() method is of type float then it rounds off the float value and returns the closest integer value. This simply means that the round() method is overloaded on the basis of the type of arguments supplied to it. Figure 4.3 illustrates the concept of method overloading as demonstrated by the round()method. For example:

```
float f = 12.5;

double d = 123.6543;

int num1 = Math.round(f);          //num1 will store 13

float num2 = Math.round(d);        //num2 will store 124.0
```

Notice that if the argument is of type double and we try to round it to an integer, then it results in an error because no such method is pre-defined which takes a double argument and rounds it to integer type.

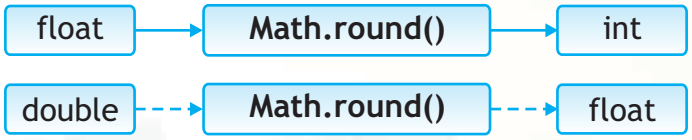

Figure 4.3 Polymorphism demonstrated by round() method

If a single argument is passed to the substring() method then it returns all characters from the start position to the end of the string whereas if two arguments are passed to the substring() method then it returns a substring of the characters between the two specified positions of the string. This simply means that the substring() method is overloaded on the basis of the number of arguments supplied to it. For example:

```
String Message = "Good Morning", Message1, Message2;

Message1 = Message.substring(5);//Message1 will store Morning

Message2 = Message.substring(0,4); //Message2 will store Good
```

Operator Overloading is another example of polymorphism. Just as methods can be overloaded, operators can also be overloaded. A very good example of operator overloading is the + operator which returns the sum of two numbers if the operands are numbers but returns the concatenated string if the operands are characters or strings. For example:

```
String A = "Hello", B = "There";
String C = A + B;              //C will store HelloThere
int num1 = 10, num2 = 20;
int num3 = num1 + num2;        //num3 will store 30
```

Abstract Class :

The classes for which objects are required to be declared are known as concrete classes like JLabel, JTextField, JComboBox etc. The classes for which it is not essential to declare objects to use them are known as abstract classes like JOptionPane. Abstract classes are normally used as base class in inheritance for which no direct object is required to be created. Also abstract classes are used for defining generic methods where there is no requirement of storing results.

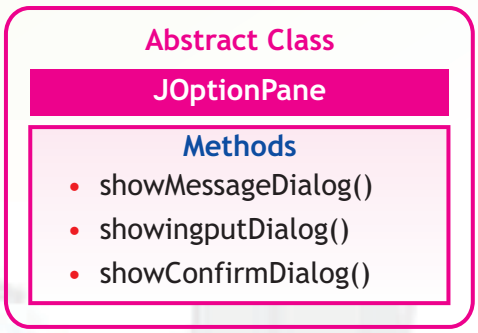

Figure 4.4
Example of an Abstract Class

Notice that the JOptionPane Class which is an abstract class has no data members and so it will not be able to store any values. The reason behind this is that because we cannot create objects of an abstract class, so we will not be able to provide any data to this class. Therefore, there is no point of having a data member.

Inheritance :

Inheritance is the most powerful feature of Object Oriented Programming, after classes themselves. Inheritance is a process of creating new class (derived class or sub class or child class) from existing class (base class or super class or parent class). The derived classes not only inherit capabilities of the base class but also can add new features of

their own. The process of Inheritance does not affect the base class. Observe the base and the derived class shown in Figure 4.5 carefully. Notice that the derived class inherits the data members namely Colour and Height from the base class and has a new data member defined in itself namely the Type. Similarly the derived class inherits the methods namely getColour() and getHeight() from the base class and has a new method defined in itself namely the setType().

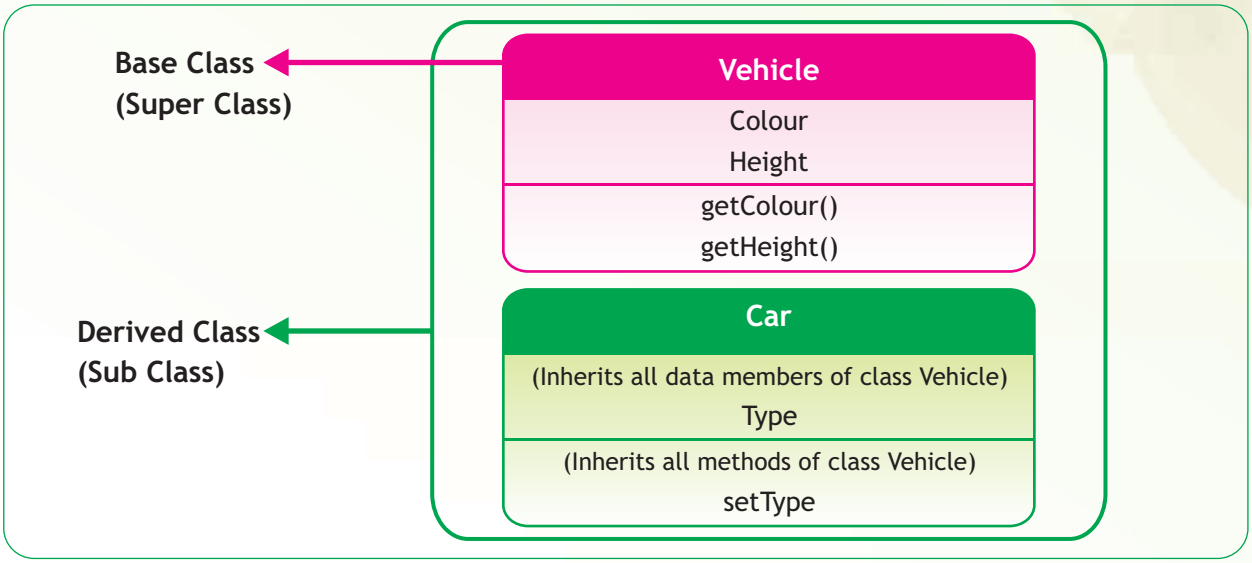

Figure 4.5 Concept of Base Class and Derived Class

The most important aspect of inheritance is that it allows reusability of code, and also a debugged class can be adapted to work in different situations. Reusability of code saves money as well as time and increases program reliability. Inheritance is very useful in original conceptualization and design of a programming problem. The idea of inheritance helps you to include features of already existing class (debugged) in a new class. Properties (data members) and methods included in super class can be invoked/accessed from the subclass. A subclass inherits all the members (data members and methods) from its superclass.

Figure 4.6 Common Examples of Inheritance

The new class can, in turn, can serve as the basis for another class definition. It is important to know that all Java objects use inheritance and every Java object can trace back up the inheritance tree to the generic class Object.

If you look at the pre-fabricated code (code automatically added by the compiler) in any of your application (JFrame Form) created in NetBeans, you will observe that javax.swing.JFrame acts as a base class (super class) and the name given by you for the JFrame Form acts as the name for the derived class (sub class). Each new JFrame Form added to the application inherits all the features of the javax.swing.JFrame class. The screen shot from one such application is shown below to illustrate the same. Here, javax.swing.JFrame is the base class and Example is the derived class.

```
public class Example extends javax.swing.JFrame
{
        /** Creates new form named Example */
        public Example()
    {
        initComponents();
    }
}
```

Figure 4.7 Sample Code Illustrating the Concept of Inheritance in Java

Notice that the extends keyword is used to inherit data members and methods of the JFrame base class for the Example class which is the derived class.

Summary

- Procedural programming paradigm focuses on breaking down a programming task into a collection of small modules known as sub routines or procedures.
- The most important distinction between Procedural and Object Oriented Programming is that where Procedural Programming paradigm uses procedures to operate on data, Object Oriented Programming paradigm bundles data and methods together and operates as an independent entity of the program.
- Class, object, data members and methods are the major components of an Object Oriented Programming paradigm.
- A class is used to encapsulate data and methods together in a single unit and an object is an instance of a class.
- Polymorphism is the ability of a method to execute in many forms.
- Method overloading and operator overloading are two examples of Polymorphism.
- The round() method of the Math class and the substring() method of the String class are good examples of method overloading.
- The + operator is a good example of operator overloading in Java.
- The classes for which it is not essential to declare objects to use them are known as abstract classes. JOptionPane is one of the examples of an abstract class.
- The concept of Inheritance allows reusability of code by including features of already existing class (called the base class) in a new class (called the derived class).
- The extends keyword is used to inherit data members and methods of the base class and allows the derived class to use these methods.

EXERCISES

MULTIPLE CHOICE QUESTIONS

1. Which of the following is not a feature of Object Oriented Programming?

   a. Inheritance b. Data Overloading

   c. Polymorphism d. Objects

2. Identify which of the following are not valid examples of method overloading?

   a. round() b. subString()

   c. subStr() d. getText()

3. Which of the following statements about the Java language is true?

   a. Both procedural and OOP are supported in Java

   b. Java supports only procedural approach

   c. Java supports only OOP approach

   d. None of the above

4. Which of the following is an Abstract class?

   a. JTextArea b. String

   c. Math d. JFrame

5. Which of the following statements is false about objects?

   a. Object is an instance of a class

   b. Object is capable of storing data

   c. Each object has its own copy of methods

   d. None of the above

6. A class can have many methods with the same name as long as the number of parameters or type of parameters is different. This OOP concept is known as

   a. Method Overriding b. Method Overloading

   c. Method Invocating d. Method Labelling

7. Which of the following statement is true?

   a. A super class is a subset of a sub class

   b. class Second extends First means First is a sub class

   c. class Second extends First means Second is a super class

   d. None of the above

8. Which feature(s) of Object Oriented Programming is illustrated in the following code snippet:

```
String A = "Hello", B = "There", C;
String C = A + B;
int num1 = 10, num2 = 20;
int num3 = num1 + num2;
C = B.substring(1);
B = A.substring(3);
```

   a. Method Overloading b. Inheritance

   c. Operator Overloading d. None of the above

ANSWER THE FOLLOWING QUESTIONS

1. Define the term Polymorphism. What are the two ways polymorphism is demonstrated in Java?
2. What is the importance of abstract classes in programming?
3. Write a short note on Operator overloading and Method Overloading citing examples for both.
4. Carefully study the code given below. It is giving an error whenever it is compiled:

```
float f = 12.5;
double d = 123.6543;
```

```
int num1 = Math.round(f);          //Statement 1

float num2 = Math.round(d);        //Statement 2

int num2 = Math.round(d);          // Statement 3
```

Identify the statement that will result in an error. Justify.

5. Carefully study the code given below:

```
String Message = "Hello! How are you?", Msg1, Msg2;

Msg1 = Message.substring(7);

Msg2 = Message.substring(0,5);
```

What will be the contents of the variables Msg1 and Msg2 after the above statements are executed?

6. Study the following code and answer the questions that follow:

```
String SMS=jTextArea1.getText();

int L=SMS.length(),Balance;

Balance=160-L;

jTextField2.setText(Integer.toString(L));

jTextField3.setText(Integer.toString(Balance));
```

a. Name any one native class of Java used in the above code.

b. Name the object created of the above mentioned native class.

c. Identify and name two methods of the native class.

d. Name the method used to convert one type of data to another and also mention the data type before and after conversion.

7. Study the following code and answer the questions that follow:

```
public class Example extends javax.swing.JFrame

{

    /** Creates new form named Example */
```

```
        public Example()
        {
            initComponents();
        }
}
```

a. Which feature of object oriented programming is depicted above?

b. Name the base class and the derived class.

c. Name the keyword used for passing on characteristics of the base class to derived class.

8. Compare and contrast the Procedural Programming paradigm and the Object Oriented Programming paradigm by writing a simple program of a mathematical calculator using both approaches. (Note: The teacher may illustrate the Procedural Programming Paradigm using any other simple programming language but should not test the students on it).

LAB EXERCISES†

1. Create a GUI application to accept a string and display it in reverse order using the substring() method.

2. Create a GUI application to create random whole numbers between 2 float numbers input by the user.

3. Create a GUI application to accept 3 numbers in separate text fields and display their sum, average, maximum or minimum after rounding the results on the click of appropriate buttons (There are four separate buttons - one for sum, one for average, one for maximum and one for minimum). The result should be displayed in the fourth text field.

4. Create a GUI application to accept the date (as 1), month (as a number like 3 for March) and year (as 2010) in separate text fields and display the date in the format: dd/mm/yy. Take care of the following points while creating the application:

† The students should be encouraged to design appropriate forms for the applications themselves.

- Verify the date input before displaying it in the suggested format and display error messages wherever applicable
- The date is accepted as 1 (for the first of any month) but should be displayed as 01 in the final format.
- The year is accepted as 2010 but displayed as 10 in the final format.

TEAM BASED TIME BOUND EXERCISES

(Team size recommended: 3 students each team)

1. Divide the class into 6 groups. Assign one class (out of JTextField, JTextArea, JLabel, JCheckbox, JRadioButton and JComboBox) to each of the groups and instruct each group to create applications demonstrating the usage of all methods learnt of that particular class. The groups may also enhance the forms using different properties of the assigned classes.
2. Divide the class into three groups and tell each group to create presentations on one of the following topics (The topics may be allocated using a draw system):
   a) Programming Paradigms
   b) The Philosophy of Object Oriented Programming
   c) Future Trends in Programming