1 Background

This document contains the assignments to be completed as part of the hands on for the subject Java Programming.

Note: All assignments in this document must be completed in the sequence in order to complete the course.

Assignment 1 : Java program with a class and method

Objective: To write and compile a java Program having a class with a method. To create an object of the class from some other class and invoke the method.

Step 1: Create a folder in D:\ drive with the name JAVA (**i.e. D:\ Java)**

Step 2: Check the basic environment variables required for working with Java by running the script created in *Assignment3*.

Step 3: Open a text editor (In Windows, notepad.exe) and type the following:

Filename Addition.java

```
/* This java file contains a class with a method to compute the
 * Sum of two numbers. This method is invoked from another class
 * by passing the necessary values as parameters
 */

class Addition{
        void fnAdd(int iFirst, int iSecond){
      int iRes = iFirst + iSecond;
                System.out.println("The output is " + iRes);
        }
}
```

Open a text editor (In Windows, notepad.exe) and type the following:

Filename ArithmeticOperation.java

```
/* This java file contains a class with the method that invokes
 * the display() method of Addition class with two parameters
 */
public class ArithmeticOperation {
        public static void main (String args[]) {
                Addition objAddition = new Addition();
                objAddition.fnAdd(3,9);
```

```
        }
}
```

Save the file as 'Summation.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac Summation.java

There is compilation error generated in the program.

Step 5: Why are we getting the error message? (Hint: Read the error message and try to find out before going to Step 6.)

Step 6: Reason for error: If there is a public class defined in the program and it contains main( ) method ,the file needs to be saved as per that class name which was not done. Resave the file as ArithmeticOperation.java and repeat Step4. If there are no syntax errors the program should compile now.

Note: Check the folder, for every class that is present in the program there is a separate .class file that gets created after compilation. It is a good practice to keep a separate .java file for each class defined.

Step 7: Run your program using the command line:

java ArithmeticOperation

Assignment 2: A class with getter and setter methods

Objective: To write a program with getter and setter methods.

Problem Description: Write a class in Java that stores the marks of 3 subjects of a student. Provide methods to set the marks and get the values of the marks. Also, provide a method called getResults(), which prints the grade of the students. Write another class that contains the main() method which creates the instance of the above created class and invokes the respective methods. The computation of grades are as follows:

Average marks	Grade
80 to 100	A
73 to 79	B+
65 to 72	B
55 to 64	C
0 to 54	D

Assignment 3: Declaring and using arrays

Objective: To write and compile a java program to deal with arrays for storing primitive data types. Depicts how to store values into the array and then retrieve the values.

Step 1: Create a folder in D:\ drive with the name JAVA (**i.e. D:\ Java)**

Step 2: Check the basic environment variables required for working with Java by running the script created in *Assignment3*.

Step 3: Open a text editor (In Windows, notepad.exe and in case of UNIX, vi) and type the following:

Filename ArrayCreation.java

```
/*
 * This java file contains a class that creates an array and uses
 * methods to set and display the values of the array
 */
public class ArrayCreation {

        int aiArray[];

        /*
         * Constructor to create an integer array that can hold
         * maximum of twelve elements
         */

        public ArrayCreation() {
                aiArray = new int[12];
        }

   /**
    * This method Stores the integers into the array using a
    * loop
    */

        public void arrayValues() {
                for(int iIndex = 0;iIndex < 12;iIndex++) {
                        aiArray[iIndex] = iIndex+1;
                }
        }

        public void arrayDisplay() {
          for(int iIndex=0;iIndex<12;iIndex++) {
```

```
                    System.out.println("aiArray[ "+ iIndex +" ] = "+
                        aiArray[iIndex] );
                    }

            }
}
```

Save the file as 'ArrayCreation.java'

Filename ArrayDemo.java

```
/*
 * This java file contains a class that creates an object of
 * ArrayCreation class and calls the method to store and display
 * the values of the array
 */
public class ArrayDemo
{
        public static void main(String args[]){
                /* Create an object of ArrayCreation class and call
            the methods */
                ArrayCreation obj ArrayCreation = new ArrayCreation();
                objArrayCreation.arrayValues();
                objArrayCreation.arrayDisplay();
        }
}
```

Save the file as 'ArrayDemo.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac ArrayDemo.java

Step 5: Run your program using the command line:

java ArrayDemo

Assignment 4: Array of objects

Objective: To write and compile a java program to deal with arrays for storing objects. Depicts how to store objects into the array and then retrieve the objects.

Step 1: Open a text editor and type the following:

Filename Book.java

```
/*
```

```
 * This java file contains a class having the attributes of a
 * book and also contains methods to set and get the properties
 * of the book the class having only getter and setter methods is
 * known as POJO [Plain Old Java Objects]/bean class
 */
public class Book{

        String bookTitle;
        float price;
        /* Method to assign and return the value for the
    datamembers of the class Book */
        public void setName(String bookName) {
                bookTitle = bookName;
        }

        public void setPrice(float cost){
                price = cost;
        }

        public String getName(){
                return bookTitle;
        }

        public float getPrice(){
                return price;
        }
}
```

Save the file as 'Book.java'

Filename BookDemo.java

```
/*
 * This java file contains a class that creates array of book
 * object and invokes the methods to set and get the properties
 * of the book
 */
class BookDemo{

        Book objBooks[] ;

        BookDemo() {
                objBooks = new Book[12];
        }

        void createBooks(){
                objBooks[0] = new Book();
                objBooks[0].setName("Gone with the wind");
```

```
                objBooks[0].setPrice(500);

                /* Create the second book object and set its
        Attributes */
                objBooks[1] = new Book();
                objBooks[1].setName("Java Programming");
                objBooks[1].setPrice(300);
          }
          void showBooks(){
            for(int iIndex=0;iIndex < objBooks.length;iIndex++) {
                System.out.println(objBooks[count].getName());
       System.out.println(objBooks[count].getPrice());
             }
          }

          public static void main(String a[]){
             BookDemo obj BookDemo = new BookDemo();
             objBookDemo.createBooks();
             objBookDemo.showBooks();
          }
}
```

Save the file as 'BookDemo.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac BookDemo.java

Step 5: Run your program using the command line:

java BookDemo

Assignment 5: Using Math class

Problem Description: Write a program in Java which generates a random number between 0 and 1. If the number generated is less than 0.5 then the program must print "The Value is less than 0.5" and if the number generated is greater than or equal to 0.5, then the program must print "The value is greater than 0.5". Write the program using a) If – else b) using the Ternary operator

Assignment 6: Using static data members and methods

Objective: To write and compile a java program to learn the concept of 'static'

Step 1: Look through the code below. Open a text editor and type the following:

Filename StaticAndNonStaic.java

```
/* This java file has a class that contains static and non static
 * instance variables and static and static and non static
 * methods to access these variables
 */

/**
 * This class contains a static variable and a non static
 * instance variable. The static variable is accessed through a
 * static method and the non static instance variable is accessed
 * through a non static method.
 */

public class StaticAndNonStatic {
 static int iStatic;
 char cChoice;

 /* Constructor. increments objectCount when an object is
  * created and sets the instance variable userChoice with the
  * parameter passed.
  */
 StaticAndNonStatic(char choice) {
    iStatic++;
    cChoice =choice;
 }

 /* The static variable objectCount is displayed using this
  * method.
  */
 static void displayObjectCount()  {
  System.out.println("Value of iStatic : "+ iStatic);
 }

 /* The non static instance variable userChoice is displayed
  * using this method.
  */
 void displayUserChoice(){
   System.out.println("The user choice is   "+ cChoice);
 }
}
```

Save the file as 'StaticAndNonStatic.java'

Filename StaticAndNonStaicDemo.java

```
/* This java file has a class that creates the object of
```

```
 * StaticAndNonStatic class and invokes the static and
 * non static methods
 */
class StaticAndNonStaticDemo  {
  public static void main(String args[])  {
    System.out.println("Before creating objects");
    StaticAndNonStatic.displayObjectCount();

    System.out.println();
    System.out.println("After creating objects");
    StaticAndNonStatic object = new StaticAndNonStatic('N');

    StaticAndNonStatic.displayObjectCount();
    StaticAndNonStatic.displayUserChoice();
  }
}
```

Save the file as 'StaticAndNonStaticDemo.java'

Assignment 7: Command line arguments

Objective: To write and compile a java program to deal with arrays for storing primitive data types. Depicts how to store values into the array and then retrieve the values.

Step 1: Open a text editor and type the following:

```
/* Java program which depicts the concept of command line
 * arguments
 */

public class CommandLineArgs  {
  public static void main(String args[])  {
          if (args.length < 2 || args.length > 2) {
                  System.out.println("Invalid no of arguments –
                    Supply  exactly two arguments");
                  System.exit(0);
          }
```

```
        System.out.println(args[0]+" "+args[1]);
 }
}
```

Save the file as 'CommandLineArgs.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac CommandLineArgs.java

Step 5: Run your program using the command line and do check the output:-

a. java CommandLineArgs
b. java CommandLineArgs Mother Teresa
c. java CommandLineArgs 5 6.8

Step 6: Make changes to the code so that the two numbers instead of being concatenated get added.

Hint: Refer to Javadoc for the wrapper classes

Assignment 8: Sorting a given set of numbers

Objective: To sort a given set of numbers

Problem Description:

- Accept five numbers from the user as command line arguments
- Sort the numbers and display the result

Programming Guidelines:

- The numbers entered could be negative or even decimal.
- The command line arguments are String, need to be converted to their numeric values
- Use any sorting technique to sort the numbers
- Display the sorted array

Assignment 9: Learning java.lang package

Objective: To learn to refer to Java documentation and use the system defined classes

Problem Description:

- Accept a number from the user as command line arguments.
- The number could be an integer or decimal value
- Display the absolute value of the input number
- Display the rounded off value of the input number
- Display the square root of the input value

Hint: There are in-built functions to achieve the above mentioned objectives. Refer to Javadoc for classes in the java.lang package

Assignment 10: Method overloading

Objective: To learn method overloading by extending a given piece of code

Problem Description:

- Compile and run the given piece of code below
- Add an overloaded method to the code that takes a double as a parameter and returns the square of it
- Write another class with the main method
- Create an object of MethodOverLoadingDemo and call the methods

```
/**
 * Program to find the square of the given number.
 */
public class MethodOverLoadingDemo{
        public int findSquare( int iNumber){
                return ( iNumber * iNumber);
        }

        //Add an overloaded method that takes a double value
        //and returns the square of it

}
```

Assignment 11: Overloading constructors

Objective: To learn the concept of constructors.

Step 1: Open a text editor and type the following:

Filename ConstructorOverload.java

```
/* This program has a class that contains multiple constructors
 * and these constructors are invoked from another class
```

```
 */
public class ConstructorOverload {

        int iNo1,iNo2;

        public ConstructorOverload(int iNumber) {
          iNo1 = iNumber;
          System.out.print("Constructor with one argument :- ");
          System.out.println("iNo1:" + iNo1 + " " +
                "iNo2:" + iNo2);
        }
        public ConstructorOverload(int iFirst,int iSecond) {
                iNo1 = iFirst;
                iNo2 = iSecond;
                System.out.print("Cons. with two arguments :- ");
                System.out.println("iNo1:" + iNo1 + " " +
                "iNo2:" + iNo2);
        }

        public ConstructorOverload() {

                System.out.print(" Default Constructor) :- ");
                System.out.println("iNo1: " + iNo1 + " " +
                "iNo2: " + iNo2);
        }
}
```

Save the file as 'ConstructorOverload.java'

Filename ConstructorDemo.java

```
/* This program will create three objects of
 * ConstructorOverlad class and invokes the constructors
 */

public class ConstructorDemo {
   public static void main(String args[]){

        /* To invoke default (with no arguments) constructor */
        ConstructorOverload obj1 = new ConstructorOverload();

        /* To Invoke a Parameterized Constructor with one
     Argument
    */
        ConstructorOverload obj2 = new ConstructorOverload(10);

        /* To Invoke a Parameterized Constructor with 2 Arguments
```

```
   */
        ConstructorOverload obj3 = new ConstructorOverload(20,30);
  }
}
```

Save the file as 'ConstructorDemo.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac ConstructorDemo.java

Step 5: Run your program using the command line:

java ConstructorDemo

Now try out the code by removing the default constructor and recompiling the code. It gives compilation error. Try to analyze the reason.

Assignment 12: Constructor chaining and dynamic polymorphism

Objective: Whenever a child class object gets created, the base class constructor gets invoked (known as constructor chaining). If the base class has a parameterized constructor then an explicit call to the base class constructor is to be made using the keyword 'super'.

Step 1: Open a text editor and type the following.

Filename Person.java

```
/* This Java file contains a Person class that depicts the
 * concept of constructors in Inheritance
 */
public class Person{
   String sName;
   int iDay,iMonth,iYear;

   public Person(String sTemp, int iTd, int iTm,  int iTy) {
     sName = sTemp;
     iDay = iTd;
     iMonth = iTm;
```

```
      iYear = iTy;
   }

   public  void displayDetails(){
            System.out.println("Name : " + sName);
    System.out.println(" DOJ :"+ iDay+"-"+ iMonth+"-"+iYear);
           }
}
```

Filename Employee.java

```
/* This Program has an Employee class which is a sub class
 * of Person class and  depicts the concept of overriding and
 * creating objects of the classes Person and Employee
 */
public class Employee extends Person {
     int iEmpNo;
     double dSalary;

    public Employee(int empId, String ename, int eday,
                    int emonth, int eyear, double sal)  {

       // Call the super class constructor
       // super(ename, eday, emonth, eyear);
      employeeId = empId;
      salary = sal;
    }

    public void displayDetails() {
      super.displayDetails();
      System.out.println("Employee Id : "+employeeId);
      System.out.println("Salary : "+salary);
    }

   public static void main(String args[])  {

     Person objectPerson =  new Person("Sanjay",10,11,1967);
     objectPerson.displayDetails();

     Employee objectEmployee =
           new Employee(111,"Dennis",21,03,2001,48000);
     objectEmployee.displayDetails();

     objectPerson = objectEmployee;
     objectPerson.displayDetails();
  }
}
```

Save the file as 'Employee.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac Employee.java

There will be compilation errors.

Step 5: Why is the error? Read the error and try to debug before going to Step6.

Step 6: Reason for error: When an object of the child class Employee is being created, the constructor of the Employee class is getting invoked. The sequence of action is, the base class constructor will get executed followed by child class constructor. In this case, the base class ie. Person has a parameterized constructor and hence default constructor will not be provided to the Person class (we learnt this in Assignment5). Hence to execute the Person class' constructor, an explicit call using super() is to be made and the parameters to be passed. There is a commented line

// **super(ename, eday, emonth, eyear);**

Remove the comment (double slash) and the line should be

super(ename,day,emont,eyear);

Now recompile the code. There are still errors.

Step 7: Why is the error? Read the error and try to debug. A weaker access privilege is being provided to the overriding function displayDetails(). Rectify the code above by making the function displayDetails() public and recompile. Now it should compile without error.

Run your program using the command line:

java Employee

Note 1: A base class reference can point to an object of the child class
(objectPerson = objectEmployee)

Note 2: When objectEmployee points to an Employee object then the Employee version of the displayDetails() function is getting called. When it is pointing to a Person object, the Person version of the displayDetails() function is getting called. So, it is being decided at run-time.

Note 3: If the base class constructor is to be invoked using the keyword super(), the call should be the first line in the child class constructor

Note 4: Note the displayDetails() method in the Employee class. A call to the displayDetails() of the Person class is being made using super.displayDetails(). When a

child class needs to access the base class version this is how it is done.

Assignment 13: Multilevel inheritance

Objective: To learn multilevel inheritance and dynamic polymorphism

Problem Description:

- Write a class called as Trainee to store GradePointAverage
- The Trainee class is a subclass of Employee
- Add a constructor to construct the Trainee class
- Add displayDetails method to display the GradePointAverage
- Write another class called as MutlilevelDemo to hold the main() method
- Create a reference to Person in main() method
- Create an object of Employee class
- Make the Person reference to point to the employee object and call the displayDetails method
- Create an object of Trainee class
- Make the Person reference to point to Trainee class and call the displayDetails method

Hint: A reference to Person class is created by including the statement

Person personReference;

Assignment 14: Abstract classes and final modifier

Objective: To understand the various modifiers used in the Java language

Problem Description:

Given the following source code, which of the commented lines amongst (1), (2), (3) or (4) could be uncommented without changing anything else in the code and without introducing errors?

```
abstract class Demo {


        protected static int count;
        private int number;
        abstract void getValues();

        final void displayValues(){
        }

        //final void compute(){ }          // (1)

}
```

```
final class MyDemo extends Demo{

		int value;
		//MyDemo(int temp){ value = temp; }	// (2)

		public static void main(String args[]) {
			Demo object = new MyDemo();
		}

		void getValues(){ }
		void compute() { }

		//void incrementCount() { count++; }	// (3)
		//void incrementNumber() { number++; }	// (4)

}
```

Analyze the results.

Assignment 15: Packages

Objective: To place a class inside a package. Compile the code and save the .class file in a different location using the –d option. See how packages affect the classpath.

Step 1: Create a directory structure \Wipro\TT\ProjectZone under your work directory **(ie.D:\Java)** and also create a folder classes under D:\Java\Classes

Step 2: Set the basic environment variables required for working with Java by running the script created in *Assignment3*

Step 3: Open a text editor and type the following:

```
/* This java files contains a class that depicts the concept of
 * packages
 */


/* Include the package statement because the class should be
  part of com.enr.MyPackage */
package Wipro.TT.ProjectZone;

public class PackageDemo  {
	public static void main(String args[]){
		System.out.println("I am part of a package now");
	}
}
```

Save the file as 'PackageDemo.java'

Note: The first line of the above code. The class PackageDemo is a part of Wipro.TT.ProjectZone now.

Step 4: Compiling the program. After compilation, the .class file got saved in the same location as the .java file. Now our objective is to save the .class file in a different location so that the class files are separated from the source code. Go to the location D:\Java\Wipro\TT\ProjectZone and compile your program using the command line:

```
javac –d  \Java\classes PackageDemo.java
```

Note: The option –d will create the directory structure if it does not exist and the .class file will automatically get created in the right path.

Step 5: Go to the location D:\Java\classes\Wipro\TT\ProjectZone and run your program using the command line:

```
java PackageDemo
```

There is a runtime error stating that the class cannot be found.

Step 6: Why is the error? Note that once a class is placed inside the package, it has to be accessed using its fully qualified name. So, how to rectify the error?

Step 7: Go to the location D:\Java\classes and run your program using the command line:

java Wipro.TT.ProjectZone.PackageDemo

Check the output of the program. **It works!!!!!**
So, whenever you are working with packages you have to be in the location one level up from where the package directory structure begins and run the code using the fully qualified name of the class.

Step 8: What if I want to run the code from any location let's say from C:\ For this, where the package directory structure is, that information has to be added to the classpath.

In Windows,
set CLASSPATH=%CLASSPATH%;d:\Java\classes;.

Now go to C:\ and use the command line:

java Wipro.TT.ProjectZone.PackageDemo

It works!!!!

Assignment 16: Access Specifiers

Objective: To place the base class in one package and the child class in some other package. How the child class needs to import the base class. It also explains how access specifiers determine what all will be available to the child class in a different package.

Step 1: Work with the same directory structure \Wipro\TT\ProjectZone under your work directory **(ie.D:\Java)** that was created in Assignment 18.

Step 2: Set the basic environment variables required for working with Java by running the script created in *Assignment3*

Step 3: Open a text editor and type the following:

```
/* This program is used to depicts different access
 * specifiers with the class declared as part of a package
 */
package Wipro.TT.ProjectZone;
public class AccessSpecifiers{
          int iDefNo;
          protected int iProNo;
          private int iPriNo;
    public AccessSpecifiers() {
        iDefNo = 10;
        iProNo = 20;
        iPriNo = 50;
   }
          public void display(){
                    System.out.println("From the public function");
                    System.out.println("iDefNo = "+iDefNo);
                    System.out.println("iProNo = "+iProNo);
       System.out.println("iPriNo = "+iPriNo);
          }
}
```

Save the file as 'AccessSpecifiers.java'. This is the base class that we have created.

Note: The AccessSpecifiers class has been given public access as it will be accessed from a different package.

Step 4: Compiling the program. Close the editor. As in Assignment18 go to the location \Java\\Wirpo\TT\ProjectZone and compile your program using the command line:

javac –d \Java\classes AccessSpecifiers.java

So, now the .class file of AccessSpecifiers is available in \Java\classes\Wipro\TT\ProjectZone

Step 5: Now it is time to write the code for the child class. Create a folder CDC7 in the location \Java\Wipro\TT. Open a text editor and type the following:

```
/* This file contains a class which is a sub class of
 * AccsessSpecifiers class and is part of different package and
 * it depicts the access specifiers across packages
 */
package Wipro.TT.CDC7;

import Wipro.TT.ProjectZone.*;

class AccessSpecifiersDemo extends AccessSpecifiers
{
        /**
         * This method accesses the instance variables of the super
    * class that are declared using different access specifiers
         */
        void view()
        {
                /* Instance variable iDefNo has default(package)
          access in super class and accessible only within
          the package, but not accessible outside the package
        */

                System.out.println(iDefNo);

                /* Instance variable iProNo has protected access in
          super class and accessible either within the
          package or within the sub class outside the
          package, but not accessible in a non sub class
          outside the package */
       System.out.println(iProNo);

                /* Instance variable iPriNo has private access in
          super class and not accessible outside the class */

          System.out.println(iPriNo);

      /* Method with public access specifier can be accessed
        in all the classes with in the package and outside
        the package */

      display();
         }
         public static void main(String a[])
```

```
        {
    AccessSpecifiersDemo object =
                        new AccessSpecifiersDemo();
                object.view();
        }


}
```

Note 1: There is an import statement in the 2nd line which makes the classes available in Package named ProjectZone to this class which is in a different package named CDC7.

Note 2: If there is a package statement and an import statement, the order has to be maintained. The package statement has to be the first line in the code.

Step 6: Compiling the program. Close the editor. As in Assignment18 go to the location \Java\Wipro\TT\CDC7 and compile your program using the command line:

javac –d \Java\classes AccessSpecifiersDemo.java

You get many compilation errors one which states that the reference of AccessSpecifiers is not available.

Step 7: Why is the error? Please remember that the code has reference to AccessSpecifiers so the classpath needs the information where the package com.enr.Package1 is. So as you did in assignment18 before, set the classpath as:

In Windows,
set CLASSPATH=%CLASSPATH%;d:\Java\MyProject\classes;.

Now try to compile the code again as before. The compilation errors are less and the errors are that iDefNo and iPriNo are not accessible

Step 8: Why is the error? Remember 'iDefNo' has package access hence it is not accessible to a different Package where the derived class is available , 'iPriNo' has private access and hence not accessible outside the ProjectZone.AccessSpecifier.

Remove these lines of code and compile.

It works!!!

Step 9: Running the code. The main method is in AccessSpecifiersDemo. So to run the code from anywhere (remember the classpath is already set), type in the command line:

java Wipro.TT.CDC7.AccessSpecifiersDemo

It should work if you have followed the steps correctly

Assignment 17: Typecasting of objects

Objective: To understand the how type casting works in the Java language

Problem Description:

Given the following source code, which of the lines (1), (2), (3) will throw compilation or run-time error? Or will the code compile and run successfully. **Try to analyze before actually compiling and running the code**

```
class Parent{

}

class Child extends Parent{

}

public class TypecastDemo{
        public static void main(String args[]){
                Parent[] arrParent;
                Child[] arrChild;

                arrParent = new Parent[10];
                arrChild = new Child[20];

                arrParent = arrChild;          //1

                arrChild = (Child[])arrParent;  //2

                arrParent = new Parent[10];
                arrChild = (Child[])arrParent;  //3
        }
}
```

Now, compile and run the code and try to analyze the results.

Assignment 18: Interfaces and reference objects

Problem Description: Given the following class definitions and declaration statements, which one of these assignment statements is legal at compile time? **Try to analyze before actually compiling the code**

```
//Class and Interface Definitions
interface MyInterface{}
class MyClass1 {

}
class MyClass2 extends MyClass1 implements MyInterface {

}
class MyClass3 implements MyInterface {

}
class ReferenceDemo{
        public static void main(String args[]){
                MyClass1 class1Object = new MyClass1 ();
                MyClass2 class2Object = new MyClass2 ();
                MyClass3 class3Object = new MyClass3();
        }
}
```

From the following options, Identify the valid option.

a) class2Object = class3Object;
b) class3Object = class2Object;
c) MyInterface InterfaceRef = class3Object;
d) class3Object = (MyClass3) class2Object;
e) class2Object = class1Object;

Hint: A base class reference can point to an object of the child class. An interface reference can also point to an object of its implementing class

Now, compile the code and try to analyze the results.

Assignment 19: Creating jar files

Objective: To place a class (which is part of a package) inside a .jar file and run the program from within the jar.

Step 1: Open a text editor and type the following:

```
/* This Java file contains a class which is part of
 * package named Wipro.TT.ProjectZone and will be placed in a jar
 * file
 */
package Wipro.TT.ProjectZone;
```

```
class Sample{
        public static void main(String args[]){
                System.out.println("I am running from inside a jar");
        }
}
```

Step 2: Save this file as Sample.java in the path **D:\Java\Wipro\TT\ProjectZone**

Step 3: Using javac –d option (as learnt in Assignment 18) compile Sample.java and save the .class file under **D:\Java\ classes.** So, Sample.class is now a part of package Wipro.TT.ProjectZone.

Step 4: Making an executable jar. The first step to create a .jar file is to create the manifest file that states which class has the main () method. The fully-qualified class name must be mentioned.

Make a text file named manifest.txt that has a single line:

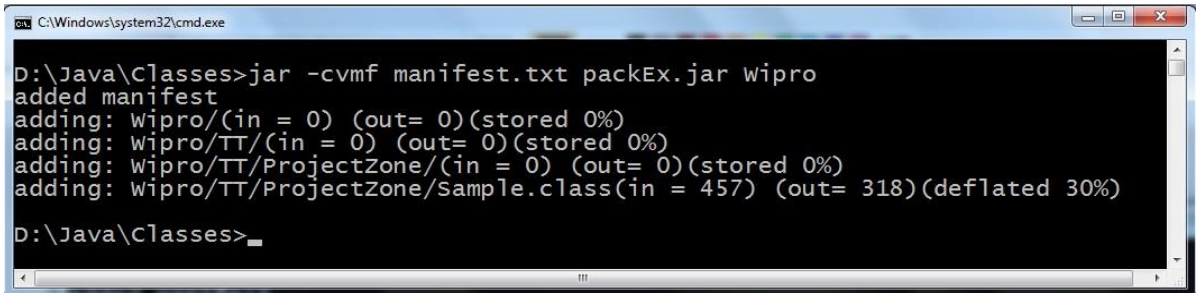
Main-Class: Wipro.TT.ProjectZone.Sample

Note 1: Give a space after the colon.
Note 2: Press return key after typing the first line

Save this manifest.txt file in the path **D:\Java\ classes**

Step 5: Go to the location **D:\Java\classes** and run the jar command from the command line as:

jar –cvmf manifest.txt packEx.jar com



Note 1: c, stands for create,
-v, verbose
-m, include information of the manifest file
-f, specify archive file name
Note 2: packEx.jar is the name of the jar file to be created.
Note 3: Only the com directory needs to be specified and the entire package will go into the JAR

Step 6: Run the code stored in the jar file
Go to the location **D:\Java\ classes** (path where the jar file has got saved) and type from the command line:

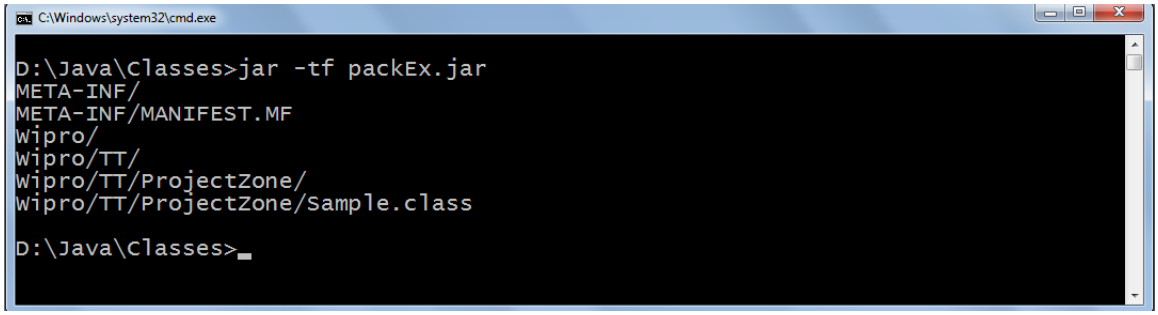
java –jar packEx.jar

Check the output of the program. **It works!!!!!**

Step 8: Let us list out the contents of packEx.jar that we just now created.
Type from the command line:

jar -tf packEx.jar

Note: -tf stands for table of files.



Note: The jar utility created the directory META-INF ('meta information') and a file MANIFEST.MF got created inside it. The contents from manifest.txt file were written into MANIFEST.MF file.

Assignment 20: Manipulation of strings

Problem Description: Write a program in Java which accepts 2 strings as command line arguments. The program should do the following:

i) print and display the total number of occurrences of the character 'w' or 'W' present in both the strings
ii) replace every occurrence of 'w' or 'W' by T
iii) convert both the strings to upper case
iv) concatenate the two strings and finally display the result

Hint: Refer to Java documentation for String and StringBuffer classes.

Assignment 21: Working with dates

Problem Description: Write a program in Java which initializes the starting date as your birthday and displays the date 90 days later in the format day name, month name, day, year (eg. Monday, April 20, 1998)

Hint: Refer to java documentation for GregorianCalendar and DateFormatter classes.

Assignment 22: Hash Table

Problem Description: Store the names of 10 major cities and the names of corresponding countries in a hash table. Accept the name of a city as a command-line argument and display the country in which it is situated. Make provision to display a message if the user either forgets to provide command-line argument or specifies a city that is not in the hash table.

Assignment 23: Working with collection classes

Problem Description: Take in 10 numbers as command line arguments and store it in a collection. The numbers are to be displayed in the reverse order in which they were entered. Proper error messages should be displayed if:

i) command line arguments have not been entered
ii) less than 10 numbers have been fed in
iii) If one of the arguments is not a valid number

Assignment 24: Exception handling

Objective: We will learn the usage of try-catch-finally blocks as a part of exception handling in Java. This assignment shows the flow of a program when an exception occurs.

Step 1: Open a text editor and type the following.

Filename: DivisionByZero.java

```
/*
 * This program is used to demonstrate the use of exception
 * handling
 */

public class DivisionByZero {
        public void division(){
                int iNumerator = 10;
                int iDenominator = 0;
                try {
```

```
        System.out.println(iNumerator + "/" + iDenominator
         + "=" + (iNumerator/iDenominator));
                    System.our.println(" After Exception ");
              }
      catch(NullPointerException e){
                     System.out.println("Null Pointer Exception");
                  }
            catch(ArithmeticException e){
                    System.out.println("Divide By Zero Error");
                  }
            finally {
                     System.out.println(" After Handling Exception ");
                  }
                  System.out.println("Happy Learning");
        }
        public static void main(String args[]){
                  new DivisionByZero().division();
                  System.out.println("End of Main ");
        }
}
```

Save the file as 'DivisionByZero.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

javac DivisionByZero.java

Step 5: Try to predict the output of the code during run-time before moving to

Step 6: Now run the code by typing in the command line:

java DivisionByZero

Analyzed the output of the program

Step 7: Make slight modification to the given code by adding a new catch block (given below) before the catch block for ArithmeticException

```
catch(Exception e){
                  System.out.println("In the exception catch block");
}
```

Step 8: Try to recompile this modified code. You will get compilation error. **Why is the error?**

Step 9: The sequence of catch blocks added to the code does matter. Since Exception class is a base class for ArithmeticException, any exception that can be caught by

ArithmeticException class can be handled by Exception class as well. So, the catch block of ArithmeticException will never get a chance to be executed and hence the compilation error.

Assignment 25: Flow of exceptions

Objective: We will see how the program flow occurs if a particular exception has not been caught.

Step 1: Open a text editor and type the following. The code is a slight modification of the code used in Assignement28.

Filename: DivisionByZero1.java

```
/*
 * This program is used to demonstrate the use of exception
 * handling
 */
public class DivisionByZero1 {
        public void division(){
                int iNumerator = 10;
                int iDenominator = 0;
                try{
        System.out.println(iNumerator + "/" + iDenominator
                 + "=" +(iNumerator/iDenominator));
                }
      catch (NullPointerException e){
                  System.out.println("Null Pointer Exception ");
                }
      finally {
                  System.out.println("End of Excep.Finally Block");
                }
                System.out.println("After Exception Handler ");
        }
        public static void main(String args[]){
                        new DivisionByZero1().division();
                System.out.println("End of Main ");
        }
}
```

Save the file as 'DivisionByZero1.java'

Step 4: Compiling the program. Close the editor. Now compile your program using the command line:

```
javac DivisionByZero1.java
```

Step 5: Try to predict the output of the code during run-time before moving to The next step.

Step 6: Now run the code by typing in the command line:

java DivisionByZero

The output of the program should come up.

Note:

a. In the changed code, the exception could not be handled by a suitable catch block.
b. However, finally block even then got executed.
c. The exception got percolated up and was finally handled by the default handler.

Make necessary changes to the code of DivisionByZero1.java so that "Returning from division" should not get printed but "Returning from main" should get printed.

Assignment 26: User defined exception

Objective: Now we will see how to code for a user defined exception class and how to throw and catch a user defined exception.

Step 1: Open a text editor and type the following. The code is a modification of the code used in Assignement28 to incorporate a user defined exception class.

```
/*
 * This program is used to demonstrate how to create and throw
 * user defined exception classes
 */

class DivisionByZeroException extends Exception{        // (1)
        public DivisionByZeroException(String message){
                super(msg);
        }
}
public class DivisionByZero2{
  public void division() throws DivisionByZeroException { //(2)
                int iNumerator = 10;
                int iDenominator = 0;
                if (iDenominator == 0)
        throw new DivisionByZeroException("Divide By Zero");
                                        //(3)
                System.out.println(iNumerator + "/" + iDenominator
              + "=" + (iNumerator/iDenominator));
                System.out.println("End of Function ");
```

```
        }
        public static void main(String args[]){
                try{
                        new DivisionByZero2().division();
                }
        catch(DivisionByZeroException e){          //(4)
                        System.out.println("Error "+ e);
                                                                                //(5)
                }
        finally {
                        System.out.println("Finally Block");
                }
                System.out.println("End of main.");
        }

}
```

Save the file as 'DivisionByZero2.java'

Step 4: Note the steps involved when coding for a user defined exception class:

1) class DivisionByZeroException is the user defined exception class which must extend from the exception class (Step marked (1))
2) The method division() declares that it **throws** exception of type DivisionByZeroException (Step marked (2))
3) The user defined exception is explicitly thrown using the throw clause (Step marked (3))
4) In the catch block in the main() method, the DivisionByZeroException is caught (Step marked (4))

Step 5: Compiling the program. Close the editor. Now compile your program using the command line:

```
javac DivisionByZero2.java
```

Step 6: Now run the code by typing the command line:

```
java  DivisionByZero2
```

The output of the program should come up.

Note: Look at output coming from the Step marked (5). Effectively .toString() method is getting invoked here. Go to javadoc and check the Exception class. See how toString() method of Object class is overridden by the Throwable class and which is used in Exception class as well (Exception class is a child class of Throwable).

Assignment 27: Debugging run-time errors

Problem Description: The code below has a Student class defined and we are trying to work with an array of Student objects, trying to set rollNo and names to the students. The code is compiling fine but throwing exception during run-time. Read the error message and rectify the code.

```
class Student{
	private int iRollNo;
	private String sName;

	public void setrollNo(int iTempRollNo){
		this.iRollNo = iTempRollNo;
	}

	public void setName(String sName){
		this.sName = sName;
	}

	public int getRollNo(){
		return iRollNo;
	}

	public String getName(){
		return sName;
	}
	public static void main(String args[]){
		Student objStudent[]=new Student[2];
		objStudent[0].setrollNo(39977);
		System.out.println(objStudent[0].getRollNo());
	}
}
```

Assignment 28: Debugging run-time errors

Problem Description: The code below compiles fine but throws run-time exception. Debug the code and rectify it. Analyze the error.

```
class RunTimeError{
	public static void main(String args[]){
		String sString= "Wipro";
		Object refObject = sString;
		String sStr = (String)refObject;
```

```
                Integer objInteger = new Integer(10);
                refObject = objInteger;
                Integer refInteger = (Integer)refObject;

                sStr = (String)refObject;

        }
}
```

Assignment 29: Reading a .properties file

Objective: To read values present in the .properties file from the java program using ResourceBundle class.

Step 1: Create a .properties file by name Trainings.properties and the contents as shown below

Filename: Trainings.properties

```
# Properties file for storing the information about products
Type=PRP,NBT,RLL,Cluster
BU=TT,WT,RLL,Competency
Location=CDC,BDC,HDC,PDC
```

Step 2: Open your favorite editor and type the code below and save the file with the name ReadProperties.java.

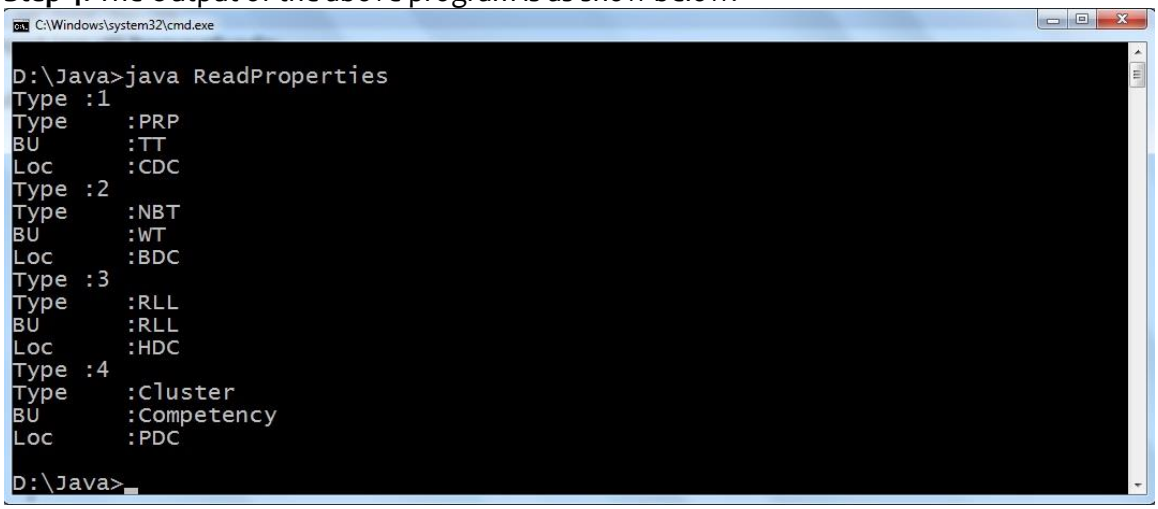
Filename: ReadProperties.java

```
/*
 * This file is used to read from a .properties file
 */

import java.util.ResourceBundle;

public class ReadProperties{

        /* Resourcebundle class object */
        ResourceBundle resourcebundle;

        public ReadProperties(){

      resourcebundle=ResourceBundle.getBundle
                        ("Trainings ");

            /* getting the key names present in the properties
```

```
        file and split w r t ',' given in the properties
        in order to get each value which is separated by
        ','*/

        String sType[]=resourcebundle.
                        getString("Type").split(",");
        String sBu[]=resourcebundle.
                        getString("BU").split(",");
                      String sLoc[]=resourcebundle.
                      getString("Location").split(",");

                    /* Loop for getting each of the values present
         in the string array */

                    for (int iIndex=0;iIndex<sType.length;iIndex++){
                            System.out.println("Type :"+(loopIndex+1));
                            System.out.println("Type       :"+sType[iIndex]);
                            System.out.println("BU         :"+sBu[iIndex]);
                            System.out.println("Loc        :"+sLoc[iIndex]);
                    }
         }
         public static void main(String[] args) {
         ReadProperties readproperties=new ReadProperties();
  }

}
```

Step 3: Compile and run the program.

Step 4: The Output of the above program is as show below:



Assignment 30: File Input and Output

Objective: To write a program that creates a FileOutputStream object, fileOutputStream, and stores bytes into the file "Data". Each byte of input is written, one at a time, to fileOutputStream. A FileInputStream object, fileInputStream, is created to read the available bytes from the file "Data" that is used as an argument to the FileInputStream constructor.

Step 1: Open a text editor and type the following.

Filename FileIOTest.java

```
/* This program demonstrates how to read and write the content
 * from the file.
 */
import java.io.*;
public class FileIOTest {
 public static void main(String[] args) throws IOException{
        FileOutputStream objFile =      new FileOutputStream("Trng");
        byte b1 = 65, b2 = 66, b3 = 67;

   objFile.write(b1);
   objFile.write(b2);
        objFile.write(b3);

        objFile.close();

        FileInputStream objFileIp = new FileInputStream("Trng");
        int iValue = objFileIp.read();
        while(iValue != -1) {
         System.out.println((byte)iValue);
         iValue = objFileIp.read();
        }
        objFileIp.close();

 }
}
```

Save the file as 'FileIOTest.java'

Note:

- **a.** After creating a FileOutputStream, each byte is written to file "Data" using FileOutputStream, one at a time.
- **b.** The stream is closed after the write operation
- **c.** A FileInputStream object, fileInputStream, is created. The read() method is invoked for fileInputStream to read available bytes of data.

Step 4: Compile and run the program

The contents of the file are printed as follows:

```
65
66
67
```

Assignment 31: Handling Primitive data types

Objective: We will see the following program which uses DataInputStream and DataOutputStream to read and write tabular data. The tabular data is formatted in columns, where each column is separated from the next by tabs. The columns contain the Employee age,name and employee number

Step 1: Open a text editor and type the following.

Filename DataIOApp.java

```
/* Program to write and read primitive types to/from a file. */
import java.io.*;

class DataIOApp{
  public static void main(String[] args) {
    try {
      DataOutputStream objDataOp =
          new DataOutputStream(
            new FileOutputStream("Trng"));    //(1)

      double[] dExp = {9.5, 9.7,3.3, 2.9, 4.9};
      int[] iEmpno = { 120, 210, 390, 480, 570 };
      String[] sName = { "Ashwaq","Jagan","Konrad","Shekar"
                  , "Gopal"};
     for(int iIndex=0;iIndex<dExp.length;iIndex++){ //(2)
      objDataOp.writeDouble(dExp[iIndex]);
      objDataOp.writeChar('\t');
      objDataOp.writeInt(iEmpno[iIndex]);
      objDataOp.writeChar('\t');
      objDataOp.writeChars(sName[Count]);
      objDataOp.writeChar('\n');
    }
    objDataOp.close();
  }
  catch (IOException E) {
    System.out.println("Error: " + E);
  }
```

```
try {
  DataInputStream objDataIp = new DataInputStream(
                new FileInputStream("Trng"));  //(3)

  double dExp;
  int iEmpno;
  String sName;
  double total = 0.0;

  try {
   while (true) {
     dExp=objDataIp.readDouble();    //(4)
     objDataIp.readChar();//throws out tab
     iEmpno = objDataIp.readInt();
     objDataIp.readChar();// throws out tab
     sName = objDataIp.readLine();
     System.out.println("Contents:"+ sName + dExp + iEmpno);
   }
  }
  catch (EOFException e) {
     System.out.println(" File Read Error : " + e);
  }
  objDataIp.close();
 }
 catch(Exception f){
   System.out.println("File Not Found : "+ f);
 }
}
}
```

Save the file as 'DataIOApp.java'

Note :

- **a.** A DataOutputStream, like other filtered output streams, must be attached to some other OutputStream. In this case, it's attached to a FileOutputStream that's set up to write to a file named Trng (marked as steps (1)
- **b.** Next, DataIOApp uses DataOutputStream's specialized write*XXX*() methods to write the invoice data (contained within arrays in the program) according to the type of data being written. (marked as step (2))
- **c.** Next, DataIOApp opens a DataInputStream on the file just written DataInputStream, also must be attached to some other InputStream. In this case, it's attached to a FileInputStream set up to read the file just written-- Trng (marked as step(3))
- **d.** DataIOTest then just reads the data back in using DataInputStream's specialized read*XXX*() methods. (marked as step(4))

Step 4: Compile and run the program.

The output of the program will give the primitive data values read form the file.

Assignment 32: Input and Output Buffering

Objective: Modify Assignment 33 such that input and output operations are performed with buffering using BufferedInputStream and BufferedOutputStream.

Assignment 33: Device Independent Input and Output

Objective: To write a program that uses similar type of interfaces to write data to a file, bytearray, standard output making use of Java's device independent IO system. It creates a FileOutputStream object, BytearrayOutputStream object and uses System.out to write data to a file "Data", bytearray and standard output.

Step 1: Open a text editor and type the following.

Filename ByteWriter.java

```
/* Prog. to demonstrate FileOutputStream, ByteArrayOutputStream,
 * System.in streams
 */
import java.io.OutputStream;
import java.io.IOException;
import java.io.ByteArrayOutputStream;
import java.io.FileOutputStream;
import java.io.FileInputStream;

public class ByteWriter{
        public void writeBytes(OutputStream outStream){
                try {
                        outStream.write(65);
                        outStream.write(66);
                        outStream.write(67);
                }
                catch(IOException e){
                  System.out.println("Error = " + e);
                }
        }
        public static void main(String [] args){
          try{
                    ByteArrayOutputStream objByteArray =
                   new  ByteteArrayOutputStream();
                    FileOutputStream objFileOp =
```

```
                    new FileOutputStream("Trng");
                     ByteWriter objByte = new ByteWriter();

                     /* To Display it on the Monitor */
objByte.writeBytes(System.out);
                    System.out.println("\n");

                    /* Write to a byte array */
                    objByte.writeBytes(objByteArray);

                    /* To Write the content to a file */
                    objByte.writeBytes(objFileOp);

        /* Printing all elements of the byte array */
                    byte bArray[] = objByteArray.toByteArray();
                    for(int iIndex=0;iIndex< bArray.length; ++iIndex){
                                System.out.println(bArray[iIndex]);
             }

                    FileInputStream objFileIp =
                                        new FileInputStream("Trng");
                    int iValue = objFileIp.read();
                    while(iValue != -1){
                            System.out.println((byte)iValue);
                            iValue = objFileIp.read();
                    }
                    objFileIp.close();
                  }
                  catch(IOException e){
                   System.out.println("Error : "+ e);
                  }
         }
}
```

Save the file as 'ByteWriter.java'

Step 4: Compile and run the program.

The output of the program is as follows:

ABC

65
66
67

65
66
67

Assignment 34: File Copy

Objective: Java can read several types of information from files: binary, Java objects, text, zipped files, etc. Here we will see how to read text files using the classes FileReader, BufferedReader, FileWriter, and BufferedWriter. It's a main program without a graphical user interface, taking parameters from the command line.

Step 1: Open a text editor and type the following.

Filename FileCopy.java

```
/* Program to demonstrate FileReader, ufferedReader, FileWriter,
 * & BufferedWriter to copy the contents of a file to another.
 */
import java.io.*;
public class FileCopy{
  public static void main(String args[]){                    //(1)
            if (args.length != 2){
              System.err.println("Usage:
                        java FileCopy file1 file2");
              System.exit(1);
            }
     File inFile  = new File(args[0]); //(2)
     File outFile = new File(args[1]);  //(3)

    try {
             BufferedReader objReader = new BufferedReader(
                new FileReader(inFile)); //(4)
             BufferedWriter objWriter = new BufferedWriter(
                New FileWriter(outFile));  //(5)

             String sStr = null;
      while ((sStr =objReader.readLine()) != null) { //(6)
       objWriter.write(sStr);     //(7)
              objWriter.newLine();
     }
           objReader.close();
           objWriter.close();
 }
 catch(IOException e){
           System.err.println(exception);
           System.exit(1);
```

```
 }
 }
}
```

Save the file as 'FileCopyDemo.java'

Note :

- **a.** This program takes two files as commandline arguments and copy the content of the first file to the second file. (marked as steps (1),(2)and (3)).
- **b.** Reader is a bufferded reader for input text file and writer is a buffered writer for output text file. They allow buffering of characters while reading and writing to enhance performance. (marked as step (4) and (5)).
- **c.** readLine() will read lines of text form the first file and writeLine() writes those lines to the second file. A loop is set up for copying all the lines form first file to second file. (marked as step(6) and (7))

Step 4: Compile the program.

Step 5: Create a text file "input.dat" with some lines of text.

Step 6: Run your program using the command line:

java FileCopy Talent.dat Training.dat

The execution of this program copy all the text lines from input.dat to output.dat

Step 7 : Open the file Training.dat. You can see the contents of Talent.dat copied to Training.dat

Assignment 35: LineNumberReader

Objective: To write a program that demonstrates the use of LineNumberReader and to print out all the lines of a file, with each line prefixed by its number.

Step 1: Open a text editor and type the following.

Filename LineNumber

```
/* Program to demonstrate the use of LineNumberReader
 * and displays the line numbers along with lines read from a
 * file */

import java.io.*;
public class LineNumber{
  public static void main(String args[]) throws IOException{
        try {
```

```
            FileReader objFileIn =
              new FileReader("Trng");     //(1)

      /* Code to Generate the line number for the read lines
            */
      LineNumberReader objLineNumber =
                new LineNumberReader(objFileIn);  //(2)
      String sStr;
      while ((sStr= objLineNumber.readLine()) != null){
        System.out.println(objLineNumber.getLineNumber() +
                  "." + sStr);        //(3)
          }
         }
   catch (IOException E) {
          System.out.println(" Error : " + E);
         }
}
}
```

Save the file as 'LineNumber.java'

Note:

- **a.** A file reader is used to read lines from a java source file.(marked as step (1).
- **b.** A filter class LineNumberReader is used to wrap the FileReader object, which helps to read the lines as well as line numbers form the connected file. (marked as step(2)).
- **c.** The getLineNumber() method is used which returns the current line number. (marked as steps (3)).

Step 4: Compile the program.

Step 5: Run your program using the command line:

java LineNumber

The lines of the java source file DataIOApp.java are printed prefixedby the numbers as follows:

1. ABC

2. 65
3. 66
4. 67
5. 65
6. 66
7. 67

Assignment 36: Buffered Reader

Objective: We will see the following program which uses BufferedReaderStream and its readLine() method to count the number of white space characters in the input.

Step 1: Open a text editor and type the following.

Filename BufferedReaderDemo

```
/* Program to Demonstrate the use BufferedReader and
 * System.in to count the number of white space characters in the
 * input.
 */
import java.io.*;
class BufferedReaderDemo{
 public static  void main(String s[]) throws IOException{
        System.out.println("Enter the String to check");

        /* Creates a filter for buffered reading from keyboard */
        BufferedReader objReader = new BufferedReader(
                        new InputStreamReader(System.in));     //(1)

        /* Reads from the Keyboard using the filter */
        String sStr= objReader.readLine();   //(2)
        System.out.println(sStr);
        int iCount=0,iLoc;

        /* To Count the number of balnk spaces */
        for(int iIndex=1; iIndex<sStr.length();iIndex++){
          iLoc=sStr.charAt(iIndex);      //(3)
          if (iLoc == 32){
                iCount++;
    }
        }
        System.out.println("No. of Blankspaces ="+ iCount);
 }
}
```

Save the file as BufferedReaderDemo.java

Note :

a. A BuffredReader is created that wraps around an InputStreamReader with a default buffer size. It reads text from a character –input stream, buffering characters so as to provide for efficient reading of characters and lines.(marked as steps (1).

b. The readLine() reads a line of text and returns the contents of the line without including any line-termination characters, or null if the end of the stream has been reached. (marked as step (2))

c. The number of white space characters are counted using a loop form the text returned by readLine() method. (marked as step(3))

Step 4: Compile and run the program.

The output of the program will give the number of white space characters in the input text form the keyboard.

Assignment 37: PushBack InputStream

Objective : To change the first character of the input stream from one case to another using pushback filter.

Problem Description : Store the text "This is Java Input/Output" into a bytearray using ByteArrayOutputStream. Using PushbackInputStream change the first character of the input stream from an uppercase 'T' to a lowercase 't'. PushbackInputStream should read the first character of the input stream and display it. It should then push back a 't' onto the stream.

Assignment 38: Object Serialization and Deserialization

Objective: We will see how to store the state of objects to a persistent storage area(file) and restore these objects using deserialization.

Step 1: Open a text editor and type the following files.

Filename Employee.java

```
/* Program to Serialize a class named Employee.*/
class Employee implements Serializable{      //(5)
  String sName;
  int iEmpNo;
  double dSalary;

  public Employee(String sName,int iEmpNo, double dSalary){
            this.sName = sName;
            this.iEmpNo = iEmpNo;
            this.dSalary = dSalary;
          }

          public String toString(){
                    return "Name= " + sName + ";Empno= "+ iEmpNo +
                ";Salary= "+dSalary;
```

```
        }
}
```

Save the file as 'Employee.java'

Filename SerializationDemo.java

```
/* Program to demonstrate how to read and write Objects to/from
 * a file.*/
import java.io.*;
public class SerializationDemo{
  public static void main(String args[]){
          try {
            Employee objEmployee = new Employee("Konrad",390,48000);

      FileOutputStream objFileOp =
              new FileOutputStream("Object.dat");  //(1)

      /* Code to store the state of the object */
            ObjectOutputStream objectOp =
           new ObjectOutputStream(fileOutputStream);  //(2)

            objectOp.writeObject(objEmployee);              //(3)
            System.out.print("Object Written to a File ");
            objectOp.flush();
            objectOp.close();
   }
   catch(Exception e){
            System.out.println("Error : " + e);
     System.exit(0);
   }

   try {
            Employee objEmp;
    /* Code to Read the Object From the file */
            FileInputStream objFileIp =
              new FileInputStream("Object.dat");  //(4)

           ObjectInputStream objectIp =
          new ObjectInputStream(fileInputStream);  //(5)

           objEmp =(Employee)objectIp.readObject();   //(6)
           System.out.println("Contents Are : " + objEmp);
           objectIp.close();
          }
          catch(Exception e){
```

```
            System.out.println("Error : " + e);
            System.exit(0);
    }
 }
}
```

Save the file as 'DemoForSerialization.java'

Note :

- **a.** A FileOutputStream is created that refers to a file named "Employee.dat" and an ObjectOutputStream is created for that file stream. (marked as steps (1) and (2)).
- **b.** The writeObject() method of ObjectOutputStream is used to serialize the Employee object. This will write the object state to the file. (marked as step (3)).
- **c.** A FileInputstream is created that refers to the file named "Employee.dat" and an ObjectInputStream is created for that file stream. (marked as step(4) and (5)).
- **d.** The readObject method of ObjectInputStream is used to deserialize the object form the file. (marked as step(6))

Step 4: Compile and run the program.

The output of the program will be as follows :

The employee details are successfully written to file named Object.dat
Name= Konrad ; Empno= 390 ; Salary= 48000

Step 5:

Open the Object.dat file form prompt and try to read the contents.
You cannot !!!
Because serialization stores the state of the objects in the binary format. By deserialization it is possible to restore the human readable format of the object states, this is done with the help of ObjectInputStream in the program.

Step 6: Modify Employee class without implementing serialization.

Compile the program again.

There is a compilation error !!

Step 7: What is the reason ?
Only an object that implements the Serialization interface can be saved and restored by the serialization facilities.

Step 8: Modify the iEmpNo instance variable of employee class to be transient.
Compile and run the program.

Step 9: What is the difference ?

The data is not saved during serialization. Why ?
Variables that are declared as transient and static are not saved during serialization.