

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/328969152>

Java Lab Manual by Asif Munir

Presentation · November 2018

CITATIONS

0

READS

1,463

1 author:



[Asif Munir](#)

BQE Software Inc

6 PUBLICATIONS 10 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Wireless Sensor Networks [View project](#)



Java Programming Lab Manual [View project](#)



Department of Computer Science & Engineering

Java Programming Lab Manual

By: Asif Munir (Assistant Prof. CSE)

Year: 3rd Yr./6th SEM

Lab Code: -6CS7A



Modern Institute of Technology & Research Centre, Alwar, Rajasthan
Affiliated to Rajasthan Technical University, Kota

Session: 2017-18(Even Sem.)

COURSE: B.TECH

YEAR/SEM: 3rd / 6th

NAME OF FACULTY: Asif Munir

Branch: CSE

INDEX

Sr. No.	Experiment Number	Page Number
1	Experiment No 1	3 -4
2	Experiment No 2	5-6
3	Experiment No 3	7-9
4	Experiment No 4	10-13
5	Experiment No 5	14-17
6	Experiment No 6	18-21
7	Experiment No 7	22-24
8	Experiment No 8	25-30
9	Experiment No 9	31-33
10	Experiment No 10	34-37

Experiment No. 1

Objective:- Write a program to check whether a number is Armstrong or not
Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

An Armstrong of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. The various examples of Armstrong numbers are 153,370,371,407.

Source Code

/* Java Program to check whether a three digit number is Armstrong or not*/

```
import java.util.Scanner;
```

```
class armstrong
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        int number=0,sum=0,rem=0,cube=0,temp=0;
```

```
        Scanner scan=new Scanner(System.in);
```

```
        System.out.println("Enter a number: ");
```

```
        number=scan.nextInt();
```

```
        System.out.println("The number entered is:"+ " " +number );
```

```
        temp=number;
```

```
        while(number!=0)
```

```
        {
```

```
            rem=number% 10;
```

```
            cube=(int)Math.pow(rem,3);
```

```
            sum=sum+cube;
```

```
            number=number/10;
```

```
        }
```

```
        if(sum==temp)
```

```
            System.out.println(temp+" "+"is an armstrong number");
```

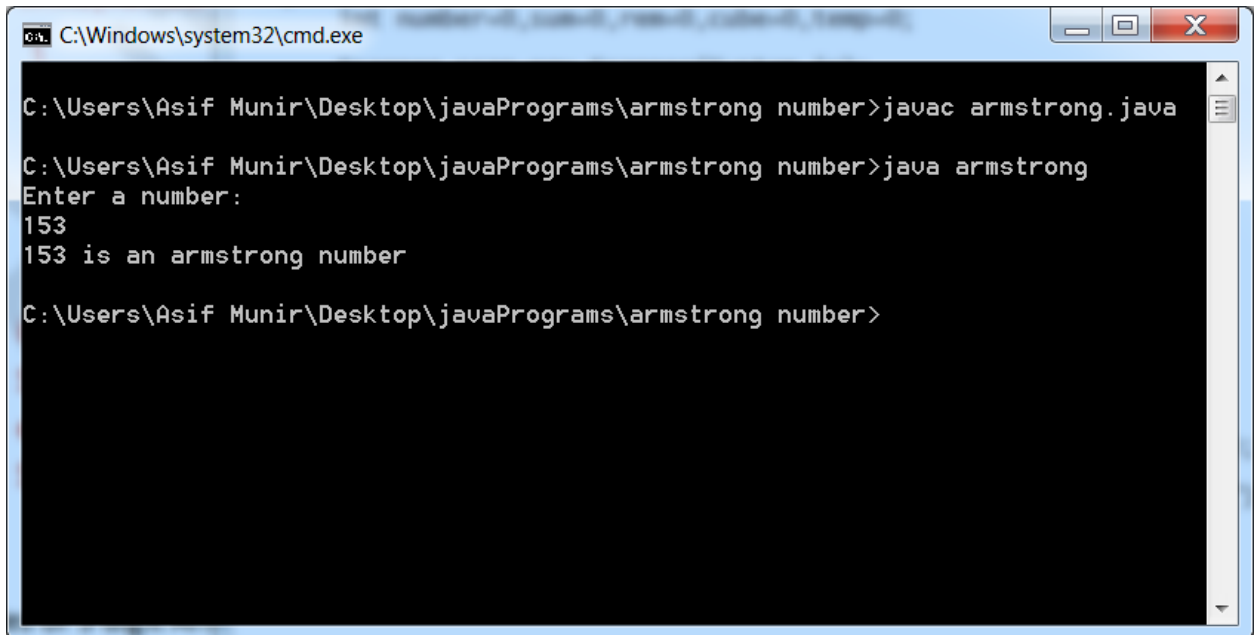
```
        else
```

```
            System.out.println(temp+" "+"is not armstrong number");
```

```
    }
```

```
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe

C:\Users\Asif Munir\Desktop\javaPrograms\armstrong number>javac armstrong.java

C:\Users\Asif Munir\Desktop\javaPrograms\armstrong number>java armstrong
Enter a number:
153
153 is an armstrong number

C:\Users\Asif Munir\Desktop\javaPrograms\armstrong number>
```

CONCLUSIONS:

So after doing the above study we have understood how to create a class and a main function in java. In addition we learned the logic of Armstrong number.

Viva questions:

1. What does static keyword mean?
2. What is an Armstrong number?
3. What is the use of import keyword
4. How is java different from C++
5. What is the use of scan keyword?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 2

Objective:- Write a program to show the concept of Constructors.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

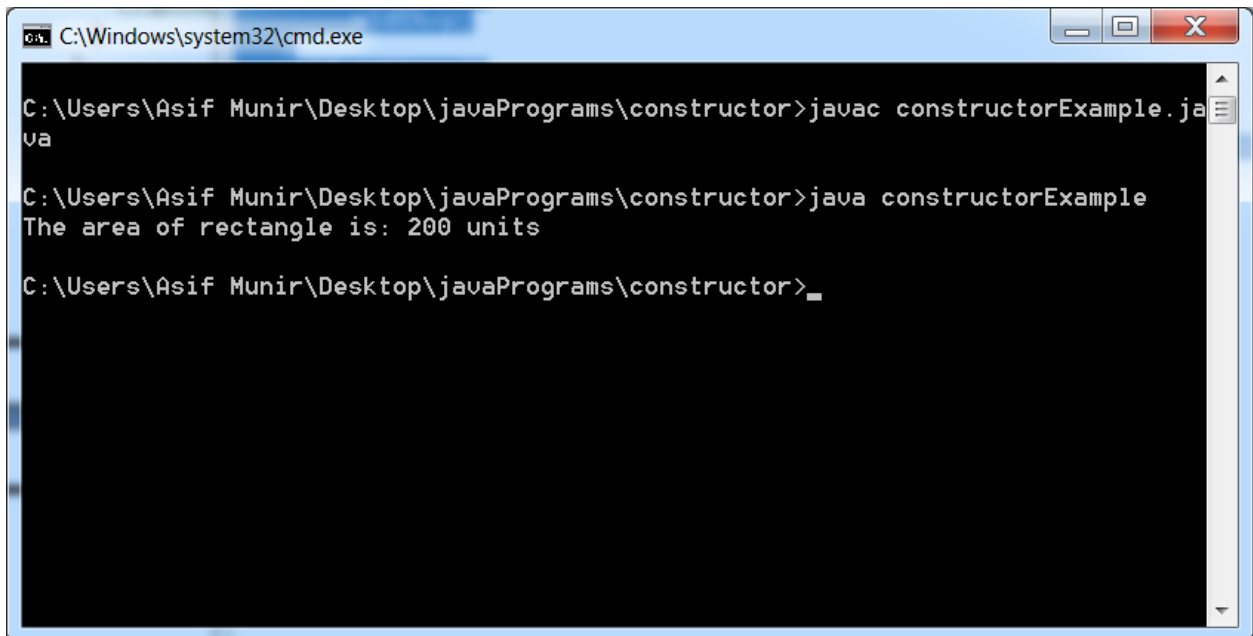
Java supports a special type of method called constructor that enables an object to initialize itself when it is created. Constructors have the same name as the class itself. Constructors do not specify a return type, not even void. This is because they return the instance of class itself.

Source Code

```
/* Java Program computing the area of rectangle illustrating constructors */
class rectangle
{
    int length, width;
    rectangle(int x, int y)
    {
        length=x;
        width=y;
    }
    void getArea()
    {
        int area=length*width;
        System.out.println("The area of rectangle is: "+ area + " units");
    }
}

class constructorExample
{
    public static void main(String args[])
    {
        rectangle r1=new rectangle(10,20);
        r1.getArea();
    }
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe

C:\Users\Asif Munir\Desktop\javaPrograms\constructor>javac constructorExample.java

C:\Users\Asif Munir\Desktop\javaPrograms\constructor>java constructorExample
The area of rectangle is: 200 units

C:\Users\Asif Munir\Desktop\javaPrograms\constructor>_
```

CONCLUSION:

So after doing the above study we have understood the concept of constructors, how to create constructor and how parameters are passed to the constructor.

Viva questions:

1. What is a constructor?
2. What is the return type of constructor?
3. What are various types of constructors?
4. What is the use of constructor?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 3

Objective:- Write a program to show the concept of method overloading.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

In java it is possible to create methods that have same name but different parameters lists and different definitions. This is called method overloading. Method overloading is used when objects are required to perform similar tasks but using different input parameters. When we call a method in an object, Java matches up the method name first, and then the number and type of parameters to decide which one of the definitions to execute. This process is known as polymorphism.

Source Code

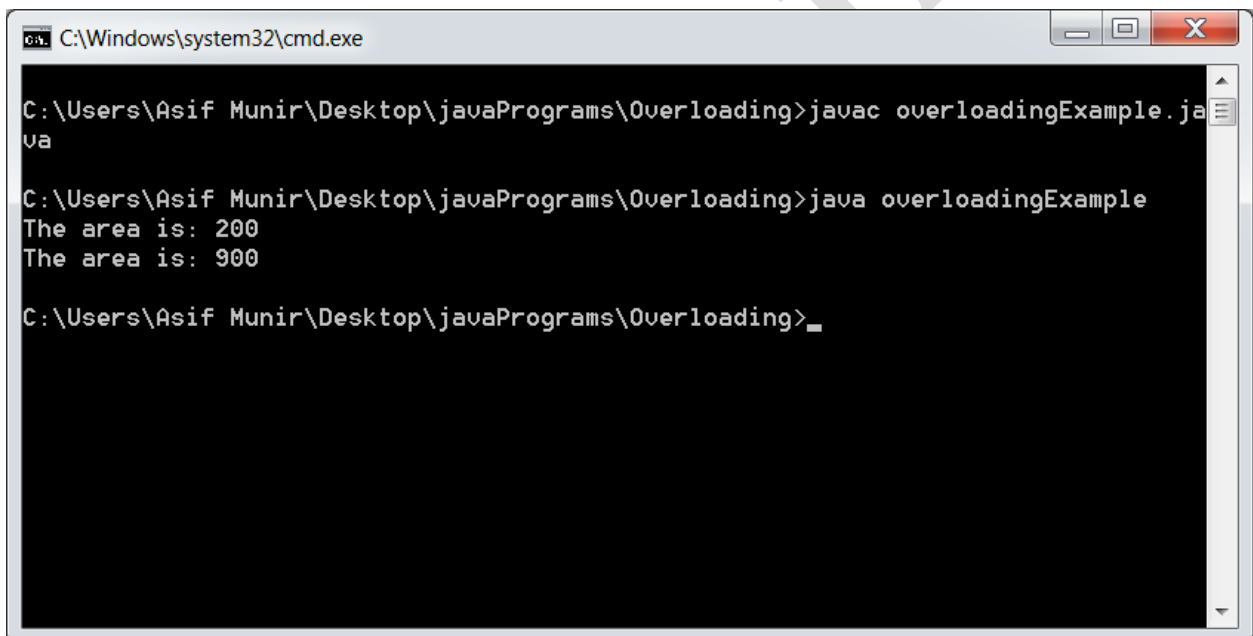
```
/* Java Program computing the area of room illustrating the concept of method overloading*/  
  
class Room  
{  
    int length;  
    int breadth;  
    Room (int x, int y)  
    {  
        length=x;  
        breadth=y;  
    }  
    Room(int x)  
    {  
        length=x;  
        breadth=x;  
    }  
    void area()  
    {  
        int x=length*breadth;
```



```
    System.out.println("The area is: "+ x);  
  }  
}
```

```
class overloadingExample  
{  
    public static void main(String args[])  
    {  
        Room r1=new Room(10,20);  
        r1.area();  
  
        Room r2=new Room(30);  
        r2.area();  
    }  
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe  
  
C:\Users\Asif Munir\Desktop\javaPrograms\Overloading>javac overloadingExample.java  
C:\Users\Asif Munir\Desktop\javaPrograms\Overloading>java overloadingExample  
The area is: 200  
The area is: 900  
C:\Users\Asif Munir\Desktop\javaPrograms\Overloading>_
```

CONCLUSION:

So after doing the above study we have understood the concept of method overloading

Viva questions:

1. What is method overloading?
2. What is polymorphism?

3. What is the use of method overloading?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 4

Objective:- Write a program to show the concept of Inheritance.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

Reusability is another aspect of OOP paradigm. Java supports this concept. Java classes can be reused in several ways. This is basically done by creating new classes, reusing the properties of existing ones. The mechanism of deriving a new class from an old one is called inheritance. The old class is known as base class or super class or parent class and the new one is called the subclass or derived class or child class. The “**extends**” keyword is used for inheritance.

The inheritance allows the subclasses to inherit all the variables and methods of their parent classes. Inheritance may take different forms:

- Single Inheritance (Only one super class)
- Multiple Inheritances (Several super classes)
- Hierarchical Inheritance (One super class, many subclasses)
- Multilevel Inheritance (Derived from a derived class)

Java does not directly implement multiple inheritances; however this concept is implemented through interface.

The diagrammatic representation of various inheritances is shown below:

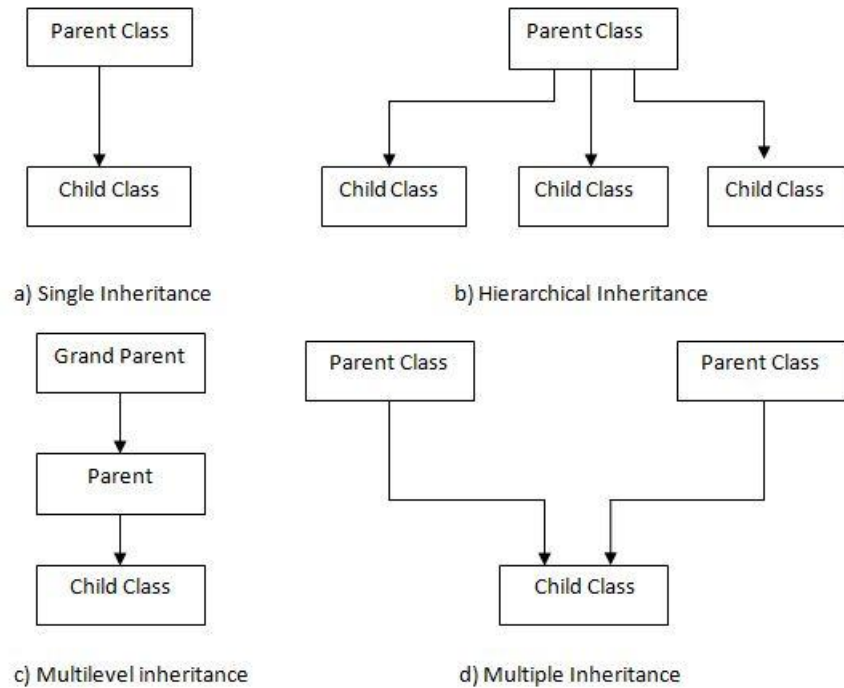


Fig 4.1: Forms of Inheritance

Source Code

/ Java Program computing the area of room illustrating the concept of single Inheritance*/*

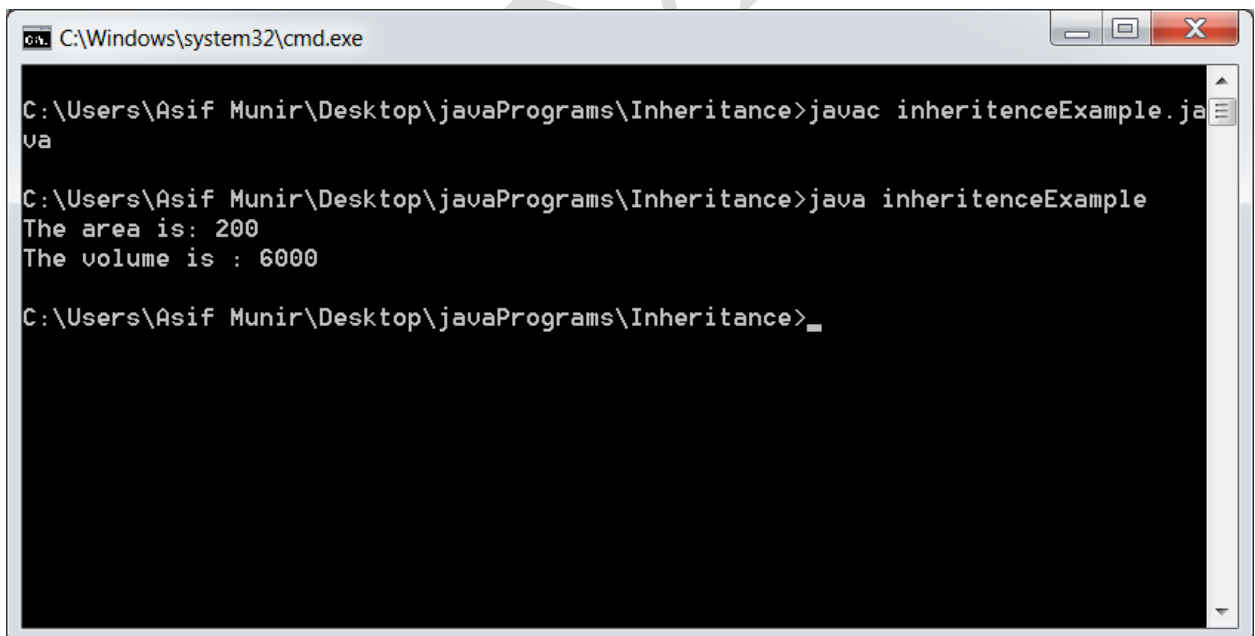
```
class Room
{
    int length;
    int breadth;
    Room(int x, int y)
    {
        length=x;
        breadth=y;
    }
    void area()
    {
        int z=length*breadth;
        System.out.println("The area is: "+z);
    }
}
```

```
class Bedroom extends Room
{
    int height;
    Bedroom(int a, int b, int c)
```

```
{
    super(a,b); //passes the value to superclass
    height=c;
}
void volume()
{
    int v=length*breadth*height;
    System.out.println("The volume is : "+v);
}
}

class inheritanceExample
{
    public static void main(String args[])
    {
        BedRoom br=new BedRoom(10,20,30);
        br.area();
        br.volume();
    }
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe

C:\Users\Asif Munir\Desktop\javaPrograms\Inheritance>javac inheritanceExample.java
C:\Users\Asif Munir\Desktop\javaPrograms\Inheritance>java inheritanceExample
The area is: 200
The volume is : 6000
C:\Users\Asif Munir\Desktop\javaPrograms\Inheritance>_
```

CONCLUSION:

So after doing the above study we have understood the concept of Inheritance.

Viva questions:

1. What is an Inheritance?

2. What is the use of inheritance?
3. What are various types of inheritance?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 5

Objective:- Write a program to show various string operations.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

Strings represent a sequence of characters. In Java strings are class objects and implemented using two classes namely String and StringBuffer. A Java string is an instantiated object of the String class. Java strings are more reliable and predictable as compared to C strings. A java character is not a character array and is not Null terminated.

The most commonly used String Methods are:

```
s2=s1.toLowerCase; //Converts String s1 to lowercase.
s2=s1.toUpperCase; //Converts String s1 to uppercase.
s2=s1.replace('x','y'); //Replaces x with y.
s1.equals(s2) //Returns true if s1=s2.
s1.equalsIgnoreCase(s2) //Returns true if s1=s2, ignoring case of character.
S1.length() //Gives length of S1
S1.charAt(n) //Gives nth character of S1
S1.compareTo(s2) //Returns -ve if s1<s2, +ve if s1>s2, zero if s1=s2
S1.concat(s2) //concatenates s1 and s2
```

Source Code

```
/* Java Program to show various string operations*/
class stringExample
{
    public static void main(String args[])
    {
        StringBuffer str=new StringBuffer("Object Oriented Language");
        System.out.println("Original String: "+str);

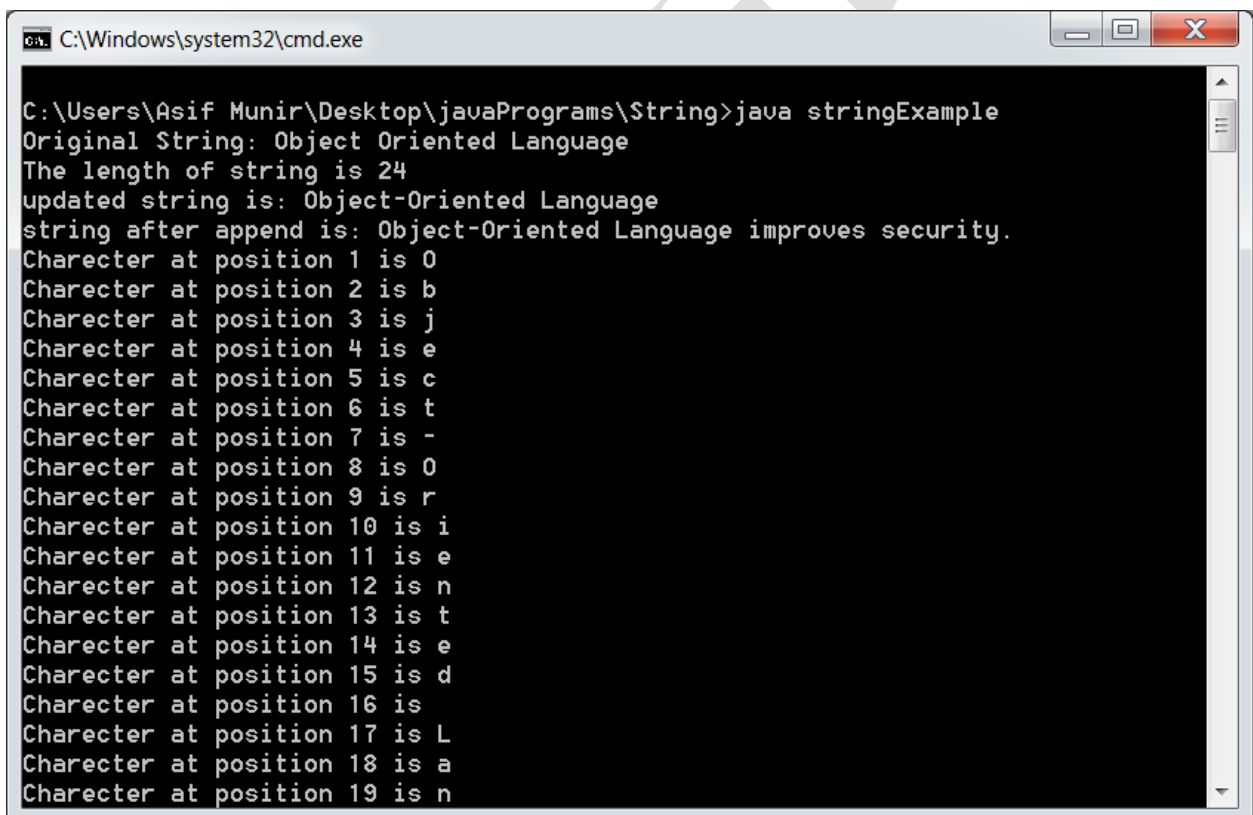
        //obtaining string length
        System.out.println("The length of string is "+ str.length());
    }
}
```

```
//Modifying Charecters
str.setCharAt(6,'-');
System.out.println("updated string is: "+str);

//Appending a string at the end.
str.append(" improves security.");
System.out.println("string after append is: "+str);

//accessing charecters in a string
for(int i=0;i<str.length();i++)
{
    System.out.println("Charecter at position "+(i+1)+" is "+str.charAt(i));
}
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe

C:\Users\Asif Munir\Desktop\javaPrograms\String>java stringExample
Original String: Object Oriented Language
The length of string is 24
updated string is: Object-Oriented Language
string after append is: Object-Oriented Language improves security.
Charecter at position 1 is O
Charecter at position 2 is b
Charecter at position 3 is j
Charecter at position 4 is e
Charecter at position 5 is c
Charecter at position 6 is t
Charecter at position 7 is -
Charecter at position 8 is O
Charecter at position 9 is r
Charecter at position 10 is i
Charecter at position 11 is e
Charecter at position 12 is n
Charecter at position 13 is t
Charecter at position 14 is e
Charecter at position 15 is d
Charecter at position 16 is 
Charecter at position 17 is L
Charecter at position 18 is a
Charecter at position 19 is n
```

Source Code

/* Java Program for Alphabetical Ordering of Strings*/

```
class stringOrdering
```



```
{
static String name[]={ "Madras","Delhi","Bombay","Hyderabad", "Ahmedabad"};

public static void main(String args[])
{

    int size=name.length;

    System.out.println("The list of Cities without String order");
    for(int i=0;i<size;i++)
    {
        System.out.println(name[i]);
    }

    String temp=null;
    for(int i=0;i<size;i++)
    {

        for(int j=i+1;j<size;j++)
        {

            if (name[j].compareTo(name[i])<0)
            {

                temp=name[i];
                name[i]=name[j];
                name[j]=temp;

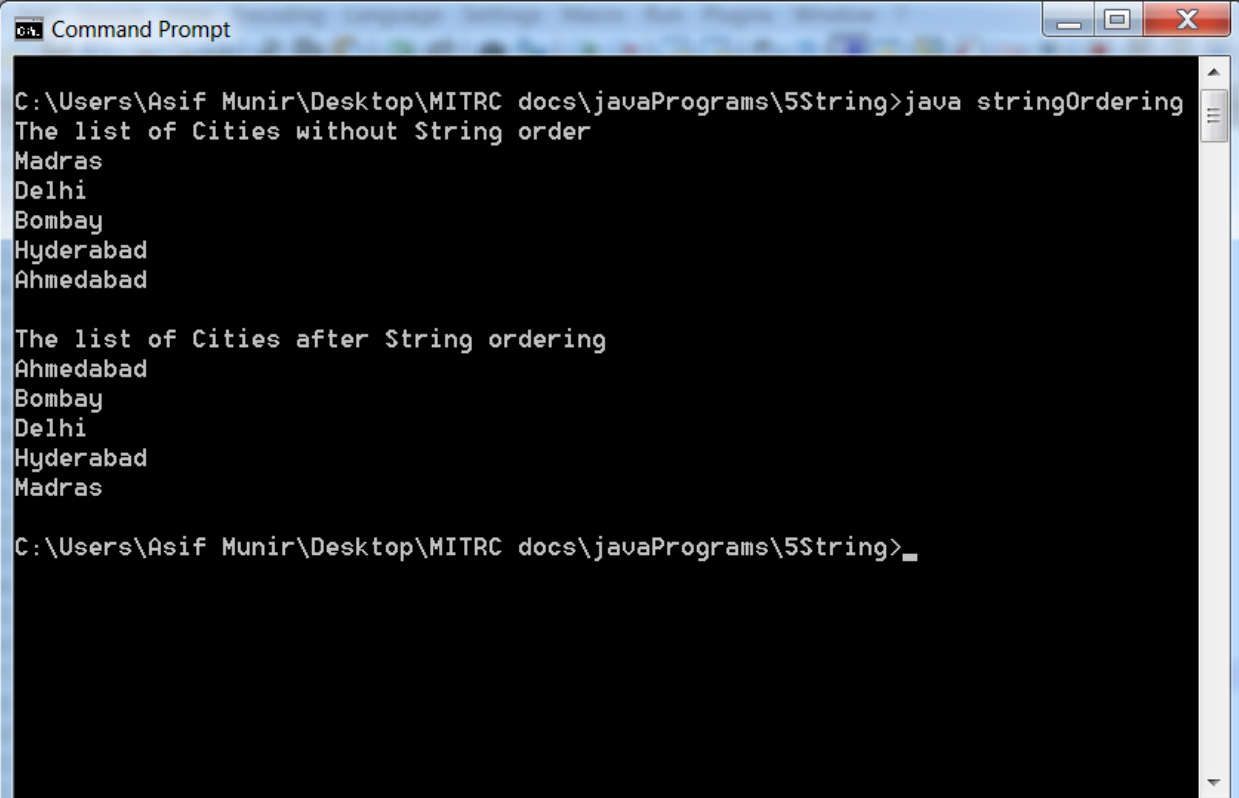
            }

        }

    }

    System.out.println("The list of Cities after String ordering ");
    for(int i=0;i<size;i++)
    {
        System.out.println(name[i]);
    }
}
```

OUTPUT:



```
Command Prompt
C:\Users\Asif Munir\Desktop\MITRC docs\javaPrograms\5String>java stringOrdering
The list of Cities without String order
Madras
Delhi
Bombay
Hyderabad
Ahmedabad

The list of Cities after String ordering
Ahmedabad
Bombay
Delhi
Hyderabad
Madras

C:\Users\Asif Munir\Desktop\MITRC docs\javaPrograms\5String>_
```

CONCLUSION:

So after doing the above study we have understood the concept and various operations on strings.

Viva questions:

1. What is a StringBuffer?
2. Which method is used for getting string length?
3. Which method is used to compare two strings?
4. Which method is used to concatenate two strings
5. Which method is used to get character at any particular location?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 6

Objective:- Write a program to show how to use interface in java.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

An interface is basically a kind of class. Like classes, interfaces contain a methods and variables but with major difference. The difference is that interface defines only abstract methods and final fields. This means that interface do not specify any code to implement these methods and data fields contain only constants. Therefore it is the responsibility of the class that implements an interface to define the code for implementation of these methods.

The syntax for defining an interface is as:

```
interface <interfaceName>
{
    Variables declaration;
    Methods declaration;
}
```

Interfaces are used as super classes. It is therefore necessary to create a class that inherits the given interface. This is done as:

```
class <classname> implements <interfaceName>
{
    Body of class
}
```

Source Code

/* Java Program calculating marks of a student using multiple inheritance implemented through interface*/

```
class student
{
    int rollNo;
    void getNumber(int n)
    {
        rollNo=n;
    }
    void putNumber()
    {
        System.out.println("Roll No is: "+ rollNo);
    }
}

class test extends student
{
    int part1, part2;
    void getMarks(int m1, int m2)
    {
        part1=m1;
        part2=m2;
    }
    void putMarks()
    {
        System.out.println("Marks obtained:");
        System.out.println("Part 1 marks = "+part1);
        System.out.println("Part 2 marks = "+part2);
    }
}

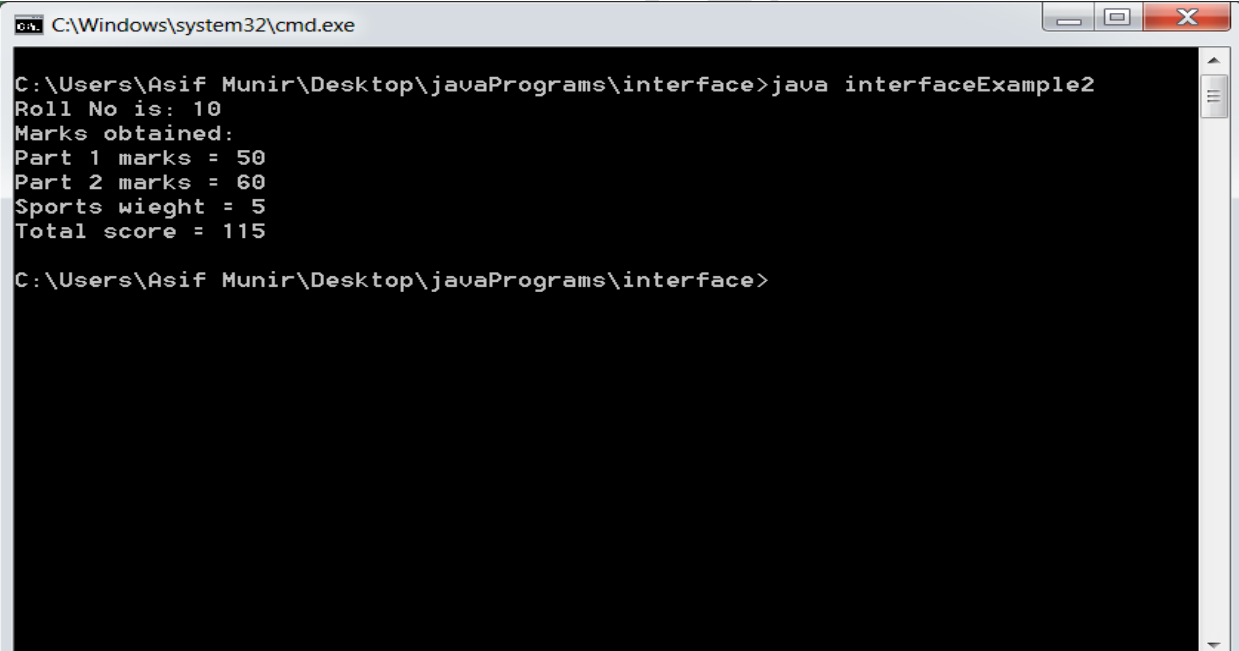
interface sports
{
    int sportsWt=5;
    void putWt();
}

class results extends test implements sports
{
    int total;
    public void putWt()
    {
        System.out.println("Sports wieght = "+ sportsWt);
    }
    void display()
}
```

```
{  
    total=part1+part2+sportsWt;  
    putNumber();  
    putMarks();  
    putWt();  
    System.out.println("Total score = "+ total);  
}  
}
```

```
class interfaceExample2  
{  
    public static void main(String args[])  
    {  
        results r1=new results();  
        r1.getNumber(10);  
        r1.getMarks(50,60);  
        r1.display();  
    }  
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe  
  
C:\Users\Asif Munir\Desktop\javaPrograms\interface>java interfaceExample2  
Roll No is: 10  
Marks obtained:  
Part 1 marks = 50  
Part 2 marks = 60  
Sports wieght = 5  
Total score = 115  
  
C:\Users\Asif Munir\Desktop\javaPrograms\interface>
```

CONCLUSION:

So after doing the above study we have understood the concept of implementing multiple inheritances through interface.

Viva questions:

1. What is a multiple inheritance?
2. What is the use of interface in java?
3. Can multiple inheritance be done without interface in java?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 7

Objective:- Write a program to show the concept of packages.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

Packages are the Java's way of grouping a variety of classes and/or interfaces together. The grouping is usually done according to functionality. In fact packages act as containers for classes. By organizing classes into packages, we achieve following benefits.

- The classes contained in packages of other programs can be easily reused.
- Packages provide a way to hide classes thus preventing other programs or packages from accessing classes that are meant for internal use only.
- Packages provide a way for separating design from coding.

Creating package involves following steps:

- Declare package at the beginning of file using the form

```
package <packageName>;
```
- Define the class that is to be put in the package and declare it public.
- Create a subdirectory under the directory where the main source files are stored. The subdirectory name must be same as package name.
- Store the listing as the classname.java file in subdirectory created.
- Compile the file. This creates .class file in the subdirectory.

Java Package can be accessed using **import** statement

Source Code

```
/* Java Program implementing packages*/

package package1;

public class classA
{
    public void displayA()
    {
        System.out.println("I am class A");
    }
}

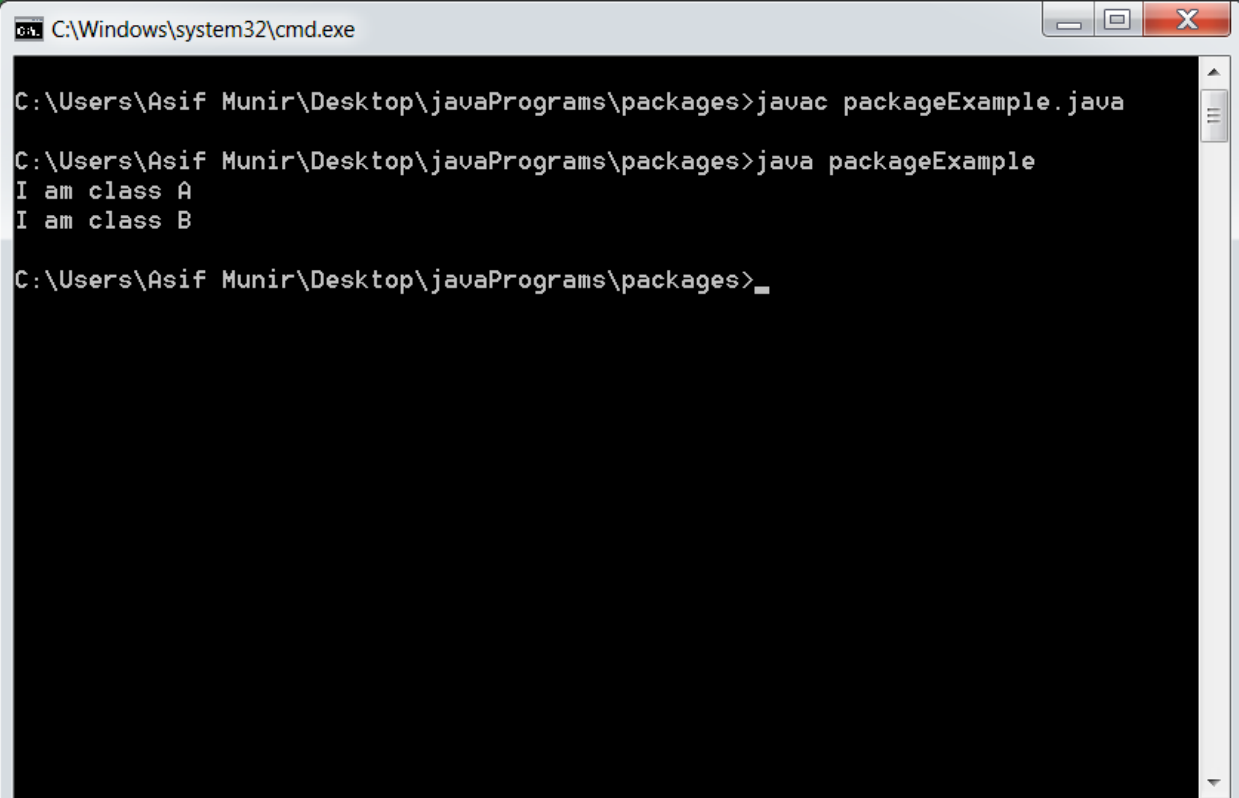
package package2;

public class classB
{
    public void displayB()
    {
        System.out.println("I am class B");
    }
}

import package1.classA;
import package2.classB;

class packageExample
{
    public static void main(String args[])
    {
        classA a=new classA();
        a.displayA();
        classB b=new classB();
        b.displayB();
    }
}
```

OUTPUT:



```
C:\Windows\system32\cmd.exe

C:\Users\Asif Munir\Desktop\javaPrograms\packages>javac packageExample.java

C:\Users\Asif Munir\Desktop\javaPrograms\packages>java packageExample
I am class A
I am class B

C:\Users\Asif Munir\Desktop\javaPrograms\packages>_
```

CONCLUSION:

So after doing the above study we have understood the concept of packages, how to create a package and how to import package.

Viva questions:

1. What is the use of package in Java?
2. How many classes can be defined in a package?
3. How does package help in making program effective?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 8

Objective:- Write a program to show the concept of threads.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

Multithreading is a conceptual programming paradigm where a program (process) is divided into two or more subprograms (processes) which can be implemented at the same time in parallel. Since the threads in java are subprograms of main application program and share same memory space, they are known as lightweight threads or lightweight processes.

Creating threads in java is simple. The run() method is the heart and soul of any thread. It makes up the entire body of a thread and is the only method in which thread's behavior can be implemented. The run method would appear as:

```
public void run()
{
    .....
}
```

We can make our class runnable by extending the class **java.lang.Thread**. The thread class can be extended as follows:

```
class <classname> extends Thread
{
    .....
}
```

Java Permits us to set the priority of thread using **setPriority()** method as follows

ThreadName.setPriority(intNumber);

The **intNumber** is an integer value to which the thread's priority is set. The thread class defines several priority constants:

MIN_PRIORITY=1
NORM_PRIORITY=5
MAX_PRIORITY=10

The life Cycle of thread is shown below:

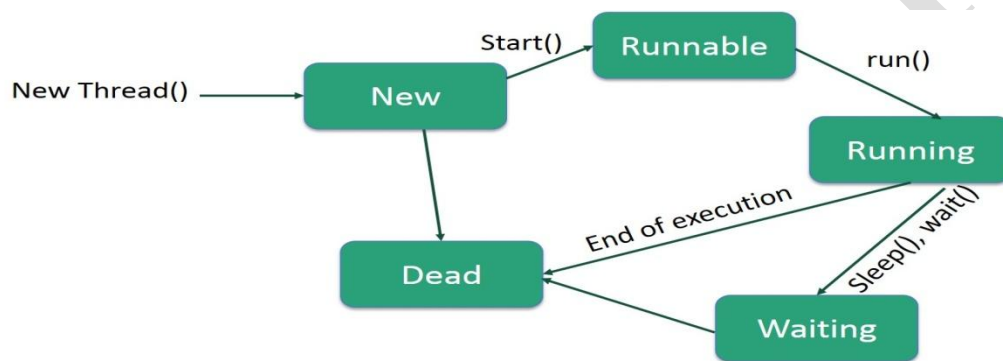


Fig 8.1: Life Cycle of Thread

Source Code

```
/* Java Program creating threads using thread classs */
class A extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("Thread A: i= "+i);
        }
        System.out.println("Exit from A");
    }
}


class B extends Thread
{
    public void run()
    {
        for(int j=1;j<=5;j++)
```

```
{
    System.out.println("Thread B: j= "+j);
}
System.out.println("Exit from B");
}
}
```

```
class C extends Thread
{
    public void run()
    {
        for(int k=1;k<=5;k++)
        {
            System.out.println("Thread C: k= "+k);
        }
        System.out.println("Exit from C");
    }
}
```

```
class threadExample1
{
    public static void main(String args[])
    {
        new A().start();
        new B().start();
        new C().start();
    }
}
```

OUTPUT:



```
Command Prompt
C:\Users\Asif Munir\Desktop\javaPrograms\8threads>java threadExample1
Thread A: i= 1
Thread A: i= 2
Thread A: i= 3
Thread A: i= 4
Thread A: i= 5
Exit from A
Thread B: j= 1
Thread B: j= 2
Thread B: j= 3
Thread B: j= 4
Thread B: j= 5
Exit from B
Thread C: k= 1
Thread C: k= 2
Thread C: k= 3
Thread C: k= 4
Thread C: k= 5
Exit from C
C:\Users\Asif Munir\Desktop\javaPrograms\8threads>_
```

Source Code

/* Java Program illustrating thread priority and yield method */

```
class A extends Thread
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {

            System.out.println("Thread A: i= "+i);
            yield();
        }
        System.out.println("Exit from A");
    }
}

class B extends Thread
{
    public void run()
    {
        for(int j=1;j<=5;j++)
        {
            System.out.println("Thread B: j= "+j);

        }
        System.out.println("Exit from B");
    }
}

class C extends Thread
{
    public void run()
    {
        for(int k=1;k<=5;k++)
        {
            System.out.println("Thread C: k= "+k);

        }
        System.out.println("Exit from C");
    }
}

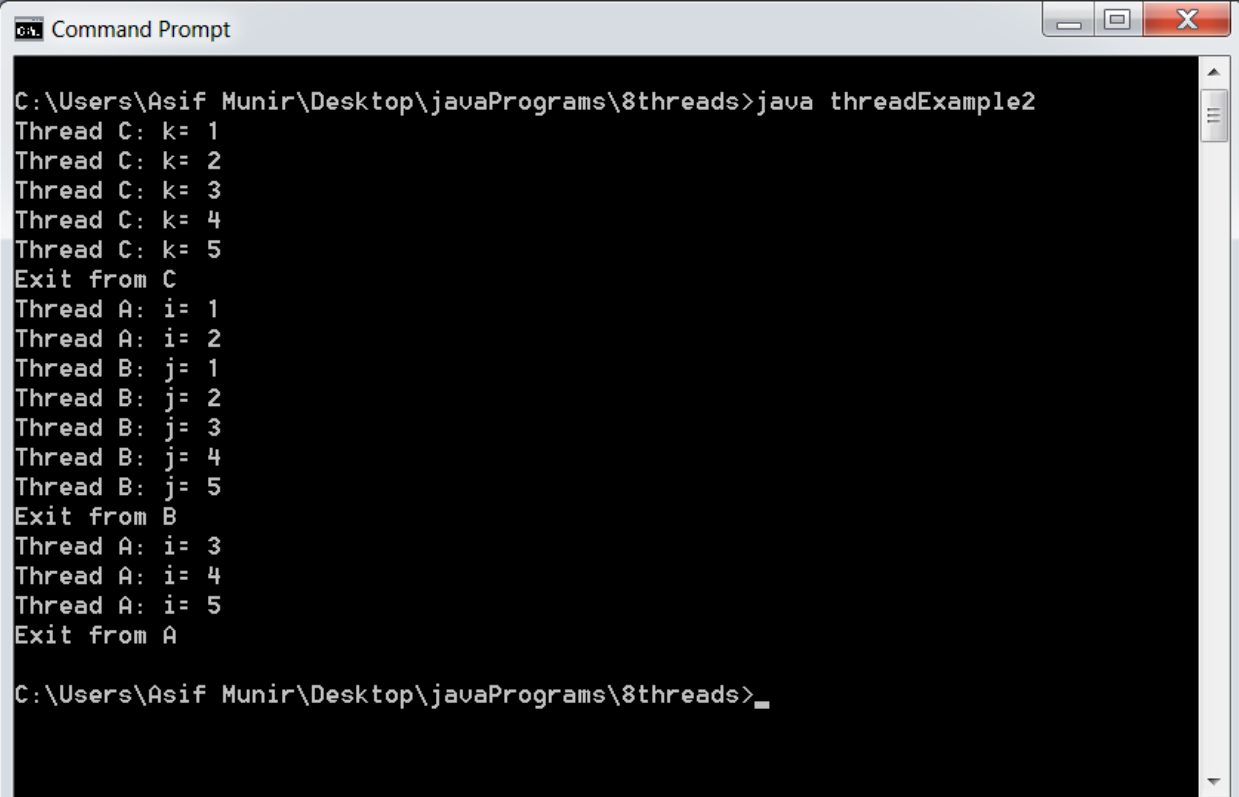
class threadExample2
{
    public static void main(String args[])
    {
    }
```

```
{
A threadA=new A();
B threadB=new B();
C threadC=new C();

threadC.setPriority(Thread.MAX_PRIORITY);
threadA.setPriority(Thread.MIN_PRIORITY);
threadB.setPriority(threadA.getPriority()+1);

threadA.start();
threadB.start();
threadC.start();
}
}
```

OUTPUT:



```
Command Prompt

C:\Users\Asif Munir\Desktop\javaPrograms\8threads>java threadExample2
Thread C: k= 1
Thread C: k= 2
Thread C: k= 3
Thread C: k= 4
Thread C: k= 5
Exit from C
Thread A: i= 1
Thread A: i= 2
Thread B: j= 1
Thread B: j= 2
Thread B: j= 3
Thread B: j= 4
Thread B: j= 5
Exit from B
Thread A: i= 3
Thread A: i= 4
Thread A: i= 5
Exit from A

C:\Users\Asif Munir\Desktop\javaPrograms\8threads>_
```

CONCLUSION:

So after doing the above study we have understood the concept of creating threads and setting thread priority.

Viva questions:

1. What is lightweight process?
2. What is the use of yield method?
3. What is use of threads?
4. How do threads help in multiprocessing?

Reference:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 9

Objective:- Write a program to show exception handling in java.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

An exception is a condition that is caused by a run time error in program. When the java interpreter encounters an error such as dividing by zero, it creates an exception object and throws it (Informs us that error has occurred). If the exception is not caught and handled properly, the interpreter will display an error message.

In Java we can use more than one catch statement for handling error. Java supports another statement known as **finally** statement that can be used to handle an exception that is not caught by any of the previous catch statement. The syntax is as:

```
try
{
.....
}
catch(....)
{
.....
}
catch(....)
{
.....
}
...
...
finally
```



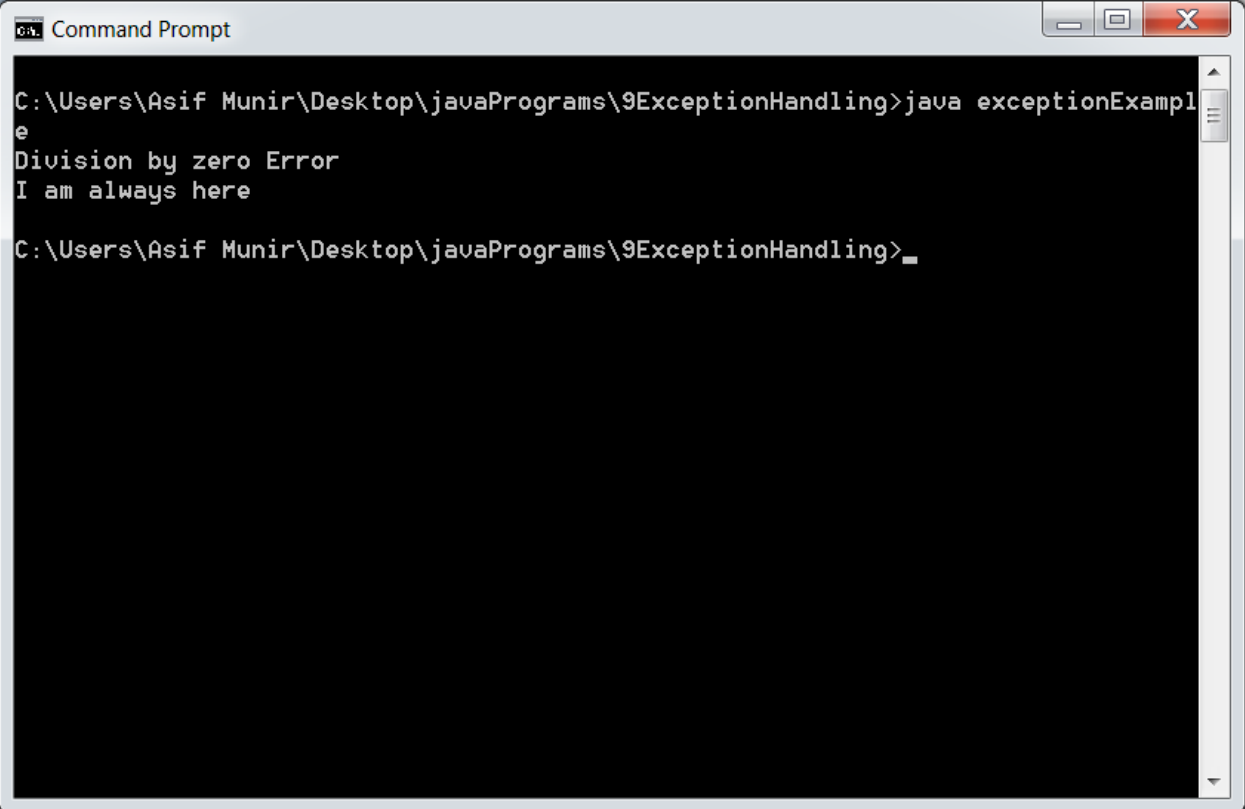
```
{  
.....  
}
```

Source Code

/* Java Program illustrating exception handling */

```
class exceptionExample  
{  
    public static void main(String args[])  
    {  
        int a[]={5,10};  
        int b=5;  
  
        try  
        {  
            int x=a[1]/0;  
        }  
  
        catch(ArithmeticException e)  
        {  
            System.out.println("Division by zero Error");  
        }  
  
        catch(ArrayIndexOutOfBoundsException e)  
        {  
            System.out.println("Array index error");  
        }  
  
        catch(ArrayStoreException e)  
        {  
            System.out.println("Wrong data type");  
        }  
  
        finally  
        {  
            System.out.println("I am always here");  
        }  
    }  
}
```

OUTPUT:



```
Command Prompt
C:\Users\Asif Munir\Desktop\javaPrograms\9ExceptionHandling>java exceptionExample
Division by zero Error
I am always here
C:\Users\Asif Munir\Desktop\javaPrograms\9ExceptionHandling>
```

CONCLUSION:

So after doing the above study we have understood the concept of exception handling by try, catch statement. In addition we have understood the concept of finally statement.

Viva questions:

1. What is the use of finally statement?
2. Difference between syntax error and logical error?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel

Experiment No. 10

Objective:- Write a program to show the concept of Applets.

Technology: Windows OS, Java Development Kit, and Notepad

Theory & Concept

Applets are small java programs that are primarily used in internet computing. They can be transported over internet from one computer to another and run using **applet Viewer**. Applets are not full featured application programs. They are usually written to accomplish small task or a component of task. Since they are usually designed for use on internet, they impose certain limitations and restrictions in their design.

- Applets do not use main() method.
- They cannot run independently. They are run from inside a web page.
- Applets can read or write files to the local computer.
- Applets cannot communicate with other servers on network.
- Applets can run any program from local computer.

The steps in developing and testing applet code are as:

1. Build an applet code (.java file)
2. Creating an executable applet (.class file)
3. Designing a web page using HTML tags. Webpage must be saved with same name as that of applet filename.
4. Preparing <APPLET> tag

5. Incorporating <APPLET> tag into the web page.
6. Creating HTML file.
7. Testing Applet code by typing below statement in cmd.

appletviewer <filename.html>

Applet code uses the services of two classes namely **Applet** and **Graphics** from java class library. Java automatically calls a series of Applet class methods for starting, running and stopping the applet code. The applet class therefore maintains the life cycle of an Applet. The **paint()** method actually displays the result of applet code on the screen.

The Applet life cycle is shown below:

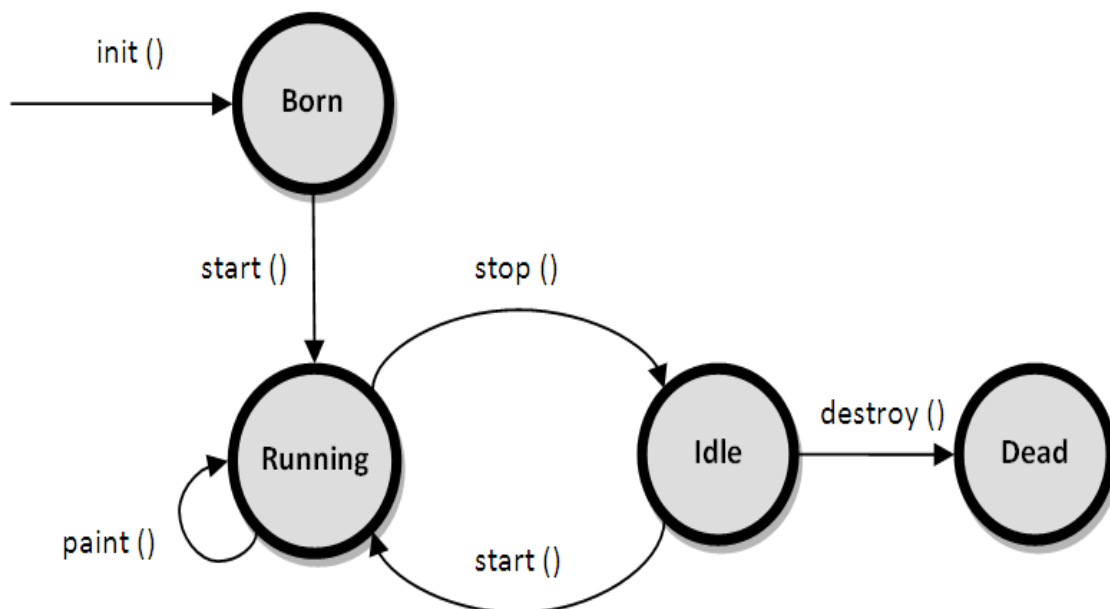


Fig 10.1: Applet Life Cycle

[Source Code](#)

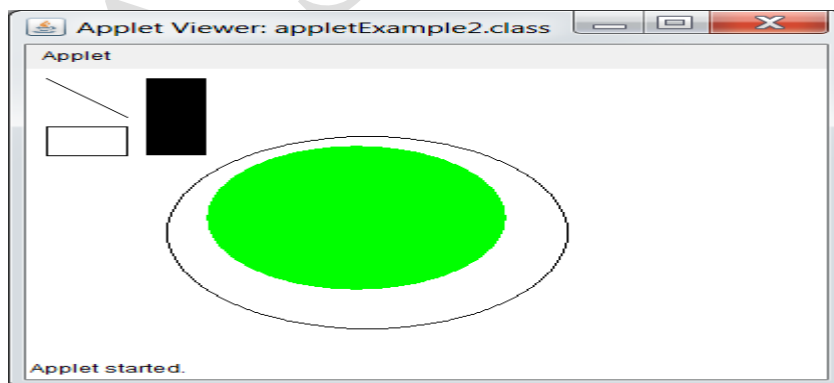
/* Applet code for drawing line, rectangle, circle and setting color */

```
import java.awt.*;
import java.applet.*;
public class appletExample2 extends Applet
{
    public void paint(Graphics g)
    {
        g.drawLine(10,10,50,50);
        g.drawRect(10,60,40,30);
        g.fillRect(60,10,30,80);
        g.drawOval(70,70,200,200);
        g.setColor(Color.green);
        g.fillOval(90,80,150,150);
    }
}
```

HTML CODE:

```
<HTML>
<HEAD>
    <H1>WELCOME TO APPLETS</H1>
</HEAD>
<BODY>
    <APPLET
        CODE=appletExample2.class
        width=400
        height=300>
    </APPLET>
</BODY>
</HTML>
```

OUTPUT:



CONCLUSION:

So after doing the above study we have understood the concept of creating applets

Viva questions:

1. What is an Applet?
2. How is Applet different from other Programs?
3. How is applet program executed?

References:

1. Java : The complete Reference by Herbertz Schildt, Oracle Press
2. Java: Black Book by Steven Holzner
3. Java: How to Program by Deitel and Deitel