Q.1(a) Write a program to calculate GCD of two numbers in java. **[5]**

(A)

```
import java.util.*;
class test
{
public static void main(String args[ ])
{
  int a, b, g, m, i;
 System.out.println("Enter nos");
 Scanner t = new Scanner (system.in);
     a = t.nextInt( );
     b = t.nextInt( );
     if (a < b)
     {
     m.a;
     }
     else
     {
     m.b;
     }
   for (i=0; i < m; i++)
   {
   if ((a % i = = 0)&&(b%i = = 0))
   {
   g=i;
    }
  }
 System.out.println("GCD ="+g);
  }
}
```

Q.1(b) Explain any three features of java. **[5]**

(A) Features of JAVA

(i) Object oriented : Java is object oriented programming language because it follows, following object oriented concept:

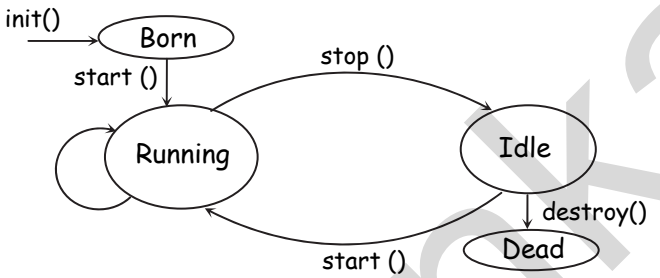
- Class & object
- Data abstraction
- Data encapsulation
- Inheritance
- Polymorphism

(b) Plat from independent : Java is platform independent, because in java, we can compile the program on one machine and we can run it on any other machine, irrespective of their operating system.

(c) Compiled and Interpreted : Java is compiled and interpreted, because it has 2 phases, compilation and interpretation.

Q.1(c) Draw and explain applet lifecycle. **[5]**

(A) Applet life cycle



The life cycle of an applet consist of following phases:

(i) **Born:** When the init() method of applet will get execute, the applet is said to in born phase.

(ii) **Running:** After initialization, appletviewer automatically calls start() method which brings the applet in running phase. In this phase, whatever is displayed in Point () method will get executed.

(iii) **Idle:** When we minimize the appletviewer or open a new window in front of it, the appletviewer automatically calls stop() method which brings the applet in idle phase.

(iv) **Dead:** When we close the appletviewer it automatically calls destroy (), which brings applet in dead phase.

Q.1(d) Explain wrapper class and its applications. **[5]**

(A) Wrapper class are special class. Which cantins different wrapper methods, which are mostly used to convert primitive data types, object and strings into each other. Following are different wrapper classes.

Premitive data types	Wrapper class
int	Integer
float	Float
double	Double
long	Long
char	Character

Wrapper classes are mainly used to perform conversing among types and objects.

Q.2(a) **1** **[5]**

1 2

1 2 3

1 2 3 4

Write a program in java to display the above pattern.

(A)

```
import java.util.*;
class test
{
public static void main (string args [ ])
{
int n = Integer.parseInt (args [0];
int i, j, a;
for (i = 1; i < n; i++)
{
  a = l;
for (j = 1; j < = i; j++)
{
   System.out.print(a);
     att;
  }
   system.out.println( );
    }
  }
}
```

Q.2(b) Explain how threads are created in java. **[5]**

(A)

- Thread can be considered as a light weight process which can be mode to execute a dedicated task.
- Multiple threads can executes simultaneously by shortly the time.

Creation of thread

Java provides following methods for creation of a thread.

Method 1 : Extending Thread class

```
 Class sample extends Thread
 {
 public void( )
 {
 ≡
  }
 }
```

```
class.test
{
public static void main (string args [ ])
 {
smaple x.new sample ( );
 }
}
```

Method 2 : Implementing Runnable Interface

```
Class sample Implements Runnable
{
public void main ()
 {
 ≡
 }
}
class test
{
 public static void main (string args[ ])
 {
      Sample x = new sample ( );
      Thread t = new Thread (x);
       x.start ( );
       }
      }
```

Q.2(c) Write a applet program to draw circle rectangle and line. **[10]**

(A)

```
import java.applet.* ;
import java.awt.* ;

/* < applet code = "shape" height = "800" width = "800 > < / applet > */
public class shape extends applet
{
    public void init( )
    {
        setBackground(color.yellow);
        setForeground(color.blue);
    }
    public void point (Graphics g)
    {
        g.drawLine(100, 100, 500, 100);
```

```
        g.drawRect(200, 200, 400, 200);
        g.drawOval(300, 500, 200, 200);
    }
}
```

Q.3(a) Write a proper example and explain the steps to create a package and then add a class or a interface. [10]

(A) **Creating a package :** Create a directory (folder) with a name same as package name, within the current working directory.

How to add class (or interface) to a Package?

Include package keyword as the first statement in a java source file. Classes defined within that file will belong to the specified package. The package statement defines the name space in which classes are stored. If package statement is omitted the classes are put in default package with no name.

Syntax : package package_name;

Steps :

1. Open a new file and add the first statement as
   package package_name;
2. Declare a public class (to be added to package).
3. Save it and compile (javac) it and ensure that the resulting.class file is available in a directory(folder) representing a package.

Suppose we have a file called HelloWorld.java, and we want to put this file in a package world. First thing we have to do is to specify the keyword package with the name of the package we want to use (world in our case) on top of our source file, before the code that defines the real classes in the package, as shown in our HelloWorld class below:

```
package world;
public class HelloWorld
{
    public void show()
    {
        System.out.println("Hello World");
    }
}
```

The above code adds class HelloWorld in package world.

Q.3(b) Write a program in java to accept the values of a, b, c and d. calculate and display ((a*d) + (b*c) / (b*d)). **[10]**

(A)

```
importjava.util.*;
class Example
{
    public static void main(String ar[])
    {
        inta,b,c,d,r;
        Scanner kbd=new Scanner(System.in);
        try
        {
            System.out.println("enter the values for a,b,c,d");
            a=kbd.nextInt();
            b=kbd.nextInt();
            c=kbd.nextInt();
            d=kbd.nextInt();
            if(b*d==0)
            throw new DenominatorZeroException();
            else
            {
                r=( (a*d)+(b*c))/(b*d);
                System.out.println("result= "+r);
            }
        }
        catch(DenominatorZeroException e)
        {
            System.out.println("denominator is zero");
        }
        catch(Exception e)
        {
            System.out.println("runtime problem");
        }
    }
}
class DenominatorZeroException extends Exception
{
}
```

Q.4(a) What is a vector? Explain any five methods of vector. [10]

(A) Vector implements a dynamic array. It is similar to ArrayList, but with two differences:

(a) Vector is synchronized.

(b) Vector contains many legacy methods that are not part of the collections framework.

Vector proves to be very useful if you don't know the size of the array in advance or you just need one that can change sizes over the lifetime of a program.

void addElement(Object element)
The object is added to the vector at the last position.

int capacity()
Returns the capacity of the vector.

int size()
Returns the size of the vector.

final object firstElement()
The method returns the first object of the invoking vector.

final Object lastElement()
The method returns the last object of the invoking vector.

final Object clone()
The method returns the duplicate of the invoking vector.

final boolean contains(object element)
The method returns true if the object is in the vector else returns false.

final intindexOf(object element)
The method returns the index of the first occurrence of the element, if the element is not present returns -1.

final intindexOf(object element, int start)
The method returns the index of first occurrence of the element at or after start. If the element is not present returns -1.

final intlastIndexOf(object element)
The method returns the index of the last occurrence of the element. If the element is not present returns -1.

final intlastIndexOf(object element, int start)
The method returns the index of last occurrence of the element at or before start. If the element is not present returns -1.

Q.4(b) Explain abstract class and abstract methods ? **[10]**

(A) Abstract Methods and Classes:

An abstract class is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be extended.

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

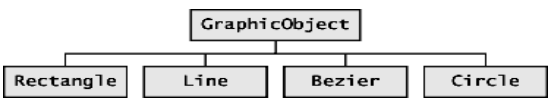
```
abstract void moveTo(double deltaX, double deltaY);
```

If a class includes abstract methods, then the class itself must be declared abstract, as in:

```
public abstract class GraphicObject {
  // declare fields
  // declare nonabstract methods
  abstract void draw();
}
```

When an abstract class is extended, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

In an object-oriented drawing application, you can draw circles, rectangles, lines, Bezier curves, and many other graphic objects. These objects all have certain states (for example: position, orientation, line color, fill color) and behaviors (for example: moveTo, rotate, resize, draw) in common. Some of these states and behaviors are the same for all graphic objects (for example: position, fill color, and moveTo). Others require different implementations (for example, resize or draw). All GraphicObjects must be able to draw or resize themselves; they just differ in how they do it. This is a perfect situation for an abstract superclass. You can take advantage of the similarities and declare all the graphic objects to inherit from the same abstract parent object (for example, GraphicObject) as shown in the following figure.



First, you declare an abstract class, GraphicObject, to provide member variables and methods that are wholly shared by all subclasses, such as the current position and the moveTo method. GraphicObject also declares abstract methods for methods, such as draw or resize, that need to be implemented by all subclasses but must be implemented in different ways. The GraphicObject class can look something like this:

```
abstract class GraphicObject
{
int x, y;
   ...
   void moveTo(intnewX, intnewY)
{
      ...
}
   abstract void draw();
   abstract void resize();
}
```

Each nonabstract subclass of GraphicObject, such as Circle and Rectangle, must provide implementations for the draw and resize methods:

```
class Circle extends GraphicObject {
   void draw() {
      ...
   }
   void resize() {
      ...
   }
}
```

Q.5(a) Write a detailed note on the following : [10]

(i) Try-catch (ii) Finally keyword

(iii) Catching multiple exception (iv) Throwing exception

(A) (i) try-catch :

- Code that can raise exception are written in **try block**
- Catch block is used to handle(catch) occurred exception of specific type mentioned in its parameter
- A single try block can have multiple catch block
- Code immediate after catch block is executed once exception occur.
- Once exception occur remaining code in try block is skipped.
- Appropriate catch block code is executed in case of multiple catch block depending on type of exception occur

(ii) finally :

- Suppose we don't want particular code to be bypassed by exception handling this problem is handled by finally keyword.
- Code written inside finally block will be executed immediately after try/catch block .
- finally block code will execute whether exception occur or not.
- It will run whether exception caught or not
- finally block can be written immediate after try block or after catch block
- The finally block is optional whereas try block need atleast one catch block or finally block

(iii) Catch multiple exception :

- A single try block can have multiple catch block, in order to raise proper exception, catch block with parent exception must be written after written after all child class exception.
- This is needed because catch statement written next to try block is checked first for execution.

(iv) Throwing Exception :

- If a method is capable of causing an exception that it does not handle ,it must specify this behavior by writing the type of exception it can throw, so caller of method will know what kind of exception called method can throw .
- If this is not handled, program will give compile time error.

Q.5(b) Write a java program to find out a number of uppercase, lowercase and blank spaces. [10]

(A)

```
import java . util . * ;
class strprocessing
{
    psrm (string args[ ])
    {
        Scanner src = new scanner (System.in) ;
        Sopln ("Enter a string") ;
        String x = src.next Line( ) ;
        int u = 0, ℓ = 0, d = 0, b = 0, s = 0 ;
        for (int i = 0 ; i < x.length( ) ; i + +)
        {
            if (x.charAt (i) > = 'A' && x.char At (i) < = 'Z')
                u + + ;
            else if (x.charAt (i) > = 'a' && x.char At (1) < = 'Z')
```

```
                ℓ + + ;
            else if (x.charAt (i) > = '0' && x.char At (i) < = 'Q')
                d + + ;
            else if (x.charAt (i) == ' ' )
                b + + ;
            else s + + ;
            Sopln ("Upper = " + u) ;
            Sopln ("Lower = " + ℓ) ;
            Sopln ("Digits = " + d) ;
            Sopln ("Blank = " + b) ;
            Sopln ("Special = " + s) ;
        }
    }
}
```

Q.6 Write short notes on **[20]**

Q.6(a) Write short note on Wrapper classes. **[5]**

(A)

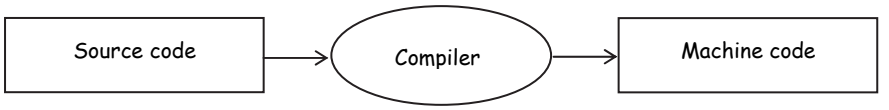
- There are certain concept in java like Vector, which can access only object and not the primitive data type.
- Hence java creates the appropriate class for all primitive data types, which are called as Wrapper class.

Primitive data type	Wrapper class
Int	Integer
float	float
char	Character
double	Double
Boolean	Boolean

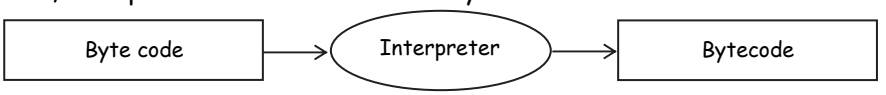
Q.6(b) Write short note on JVM. **[5]**

(A)

- It stands for java virtual machine
- Java is compiled and interpreted programming language
- Java has 2 phases ie. Compilation and interpretation
- Compilation is done by compiler whereas interpretation is done by interpreter.
- In Java, compiler converts source code into a special code called as byte code.



whereas, interpreter is used to convert bytecode into machine code



Java interpreter is called as JVM which convert bytecode into machine code to make java as platform independent.

Q.6(c) Write short note on Abstract methods and classes. [5]

(A)

- It is something exactly opposite to that of final method. Abstract methods are those methods which we need to compulsory override in its derived class.
- If we do not override abstract method, compiler will generate error
- Abstract method normally only declared in the base class and get defined in the derived class.
- If a class contains one or more abstract method, then the class is called abstract class.
- We cannot create object of the abstract class.

e.g.
```
abstract class A
{
abstract void display( );
=
}
class B extends A
{
void display ()
{
=
}
}
```

Q.6(d) Write short note on Package. [5]

(A)

- Package is a collection of similar classes.
- The various advantages of packages is as follows :

(1) Organizing Classes :

› Classes can be grouped together into packages so it becomes easy to retrieve.

(2) Improve Hiding :

› Classes kept in packages are hidden from outside world.

› When package is imported, then only public classes can be accessed.

(3) Avoiding Naming Conflict :

- Two classes with same name cannot be kept in same package but can be kept in different packages.
- Such packages can be referred with the help of their respective package name.

(4) Helps in Reusability :

- Classes kept in packages can be easily reused by importing that package.
- Hence it helps in reusability of existing code.

❑ ❑ ❑ ❑ ❑