Name: Bilal Khondannavar **Class: BCA V**

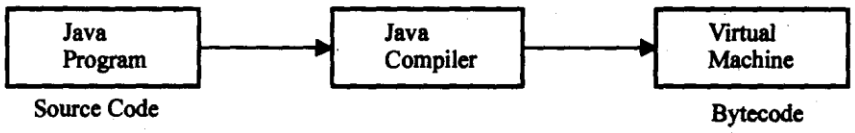
Internet Programming- 2015

1. a) What are the features of java ? Explain them briefly. (2016- 1B)

b) What is JVM? Explain it.

All language compilers translate source code into *machine code* for a specific computer. Java compiler also does the same thing. Then, how does Java achieve architecture neutrality? The answer is that the Java compiler produces an intermedia code known as bytecode for an machine that does not exist. This machine is called the *Java Virtual Machine* and it exists only inside the computer memory. It is a simulated computer within the computer and does all major functions of a real computer. Figure 3.6 illustrates the process of compiling a Java program into bytecode which is also referred to as *virtual machine code.*

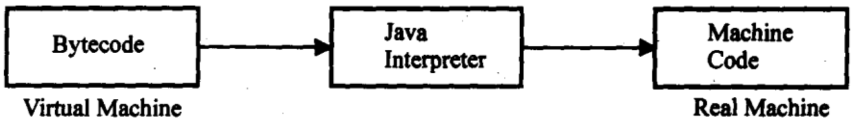
Fig. 3.6



Process of compilation

The virtual machine code is not machine specific. The machine specific code (known as machine code) is generated by the Java interpreter by acting as an intermediary between the virtual machine and the real machine as shown in Fig. 3.7. Remember that the interpreter is different for different machines.

Fig. 3.7



Process of converting bytecode into machine code

c) Write the program structure of java. (2016- 1A)

2. a) Explain briefly the life cycle of a thread. (2016- 6A)

b) Write a java program to demonstrate mouse events.

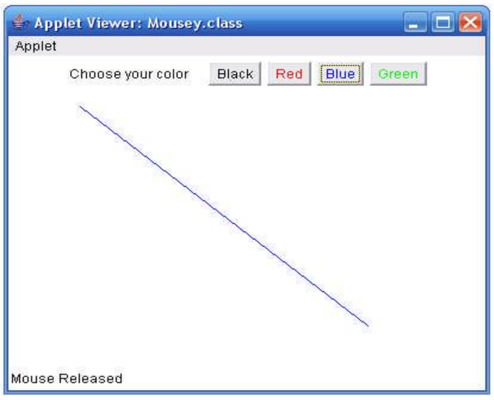
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*<applet code="Mousey.class" height=300 width=400></applet>*/
public class Mousey extends Applet implements MouseListener,ActionListener
{
int x1,x2,y1,y2;
Button r,g,b,bl;
Label l1;
String s,msg;
public void init()
```

```
{
l1=new Label("Choose your color");
r=new Button("Red");
g=new Button("Green");
b=new Button("Black");
bl=new Button("Blue");
r.setForeground(Color.red);
g.setForeground(Color.green);
b.setForeground(Color.black);
bl.setForeground(Color.blue);
add(l1);
add(b);
add(r);
add(bl);
add(g);
r.addActionListener(this);
g.addActionListener(this);
b.addActionListener(this);
bl.addActionListener(this);
addMouseListener(this);
}

public void mousePressed(MouseEvent e)
{
x1=e.getX();
y1=e.getY();
msg="Mouse Pressed";
}
public void mouseReleased(MouseEvent e)
{
x2=e.getX();
y2=e.getY();
repaint( );
msg="Mouse Released";
}
public void mouseClicked(MouseEvent e)
{
msg="Mouse Clicked";
}
public void mouseExited(MouseEvent e)
{
msg="Mouse Exited";
}
public void mouseEntered(MouseEvent e)
{
msg="Mouse Entered";
}
public void paint(Graphics g)
```

```
{
if(s=="red")
g.setColor(Color.red);
if(s=="black")
g.setColor(Color.black);
if(s=="blue")
g.setColor(Color.blue);
if(s=="green")
g.setColor(Color.green);
g.drawLine(x1,y1,x2,y2);
showStatus(msg);
}

public void actionPerformed(ActionEvent ae)
{
if(ae.getSource()==r)
s="red";
if(ae.getSource()==g)
s="green";
if(ae.getSource()==b)
s="black";
if(ae.getSource()==bl)
s="blue";
}
}
```

Output

```
E:\Programs>javac Mousey.java
E:\Programs>appletviewer Mousey.java
```



3. a) What are the mathematical functions? Explain the briefly.

Mathematical functions such as cos, sqrt, log, etc. are frequently used in analysis of real life problems. Java supports these basic math functions through Math class defined in the java.lang package. Table 5.12 lists the math functions defined in the Math class. These functions should be used as follows:

Math. `function_name ( )`

Function	Action

sin (x)	Returns the sine of the angle x in radians
cos(x)	Returns the cosine of the angle x in radians
tan (x)	Returns the cosine of the angle x in radians
asin(y)	Returns the angle whose sine is y
acos(y)	Returns the angle whose cosine is y
atan(y)	Returns the angle whose tangent is y
atan2(x,y)	Returns the angle whose tangent is x/y
pow(x,y)	Returns x raised to y (xr)
exp(x)	Returns e raised to x (ex)
log (x)	Returns the natural logarithm of x
sqrt(x)	Returns the square root of x
ceil (x)	Returns the smallest whole number greater than or equal to x. (Rounding up)
iloor(x)	Returns the largest whole number less than or equal to x (Rounded down)
rint(x)	Returns the truncated value of x.
abs(a)	Returns the absolute value of a
max(a,b)	Returns the maximum of a and b
min(a,b)	Returns the minimum of a and b

b) What is narrowing conversion? Explain with example.

The narrowing conversion occurs from a type to a different type that has a smaller size, such as from a long (64 bits) to an int (32 bits).

In general, the narrowing primitive conversion can occur in these cases:

1. short to byte or char
2. char to byte or short
3. int to byte, short, or char
4. long to byte, short, or char
5. float to byte, short, char, int, or long
6. double to byte, short, char, int, long, or float

- The narrowing primitive conversion must be explicit.
- You need to specify the target type in parentheses.

```
public class MainClass

{

public static void main(String[] args)

{

   long a = 10;

   int b = (int) a; // narrowing conversion


   System.out.println(a);
   System.out.println(b);
  }
}
```

c) What is java API packages?

Java system provides a large number of classes grouped into different packages according to functionality. Most of the time we use the packages available with the Java system.
Java System Packages and Their Classes

Package name	**Contents**
java.lang	Language support classes. These are classes that Java compiler itself uses and therefore they are automatically imported. They include classes for primitive types, strings, math functions, threads and exceptions.
java.util	utility classes such as vectors, hash tables. random numbers, date, etc.
java.io	Input/output support classes. They provide facilities for the input and output of data.
java.awt	Set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.
java.net	Classes for networking. They include classes for communicating with local computers as well as with internet servers.
java.applet	Classes for creating and implementing applets.

4. a) What is exception? And write a java program to demonstrate User defined exceptions.

An *exception* is a condition that is caused by a run-time error in the program. When the Java interpreter encounters an error such as dividing an integer by zero, it creates an exception object and throws it(i.e., informs us that an error has occurred).

```
/* Program to find Sort list of elements in Ascending and Descending orderand show
Exception Handling Mechanism*/
import java.io.*;
class ExpDemo
{
public static void main(String args[])
{
try
{
DataInputStream di=new DataInputStream(System.in);
System.out.println("Enter limit");
int n=Integer.parseInt(di.readLine());
int i;
int a[]=new int[50];
System.out.println("Enter array elements");
for(i=0;i<n;i++)
{
try{
a[i]=Integer.parseInt(di.readLine());
}
catch(NumberFormatException e)
{
System.out.println("Numbers only....!!!");
i--;
}
}
System.out.println("The array elements are");
for(i=0;i<n;i++)
System.out.println(a[i]);
for(i=0;i<n;i++)
{
for(int j=i+1;j<n;j++)
{
if(a[i]>a[j])
{
int temp=a[i];
a[i]=a[j]; a[j]=temp;
}
}
}
System.out.println("The Sorted elements in ascending order are");
for(i=0;i<n;i++)
System.out.println(a[i]);
System.out.println("\nThe Sorted elements in descending order are");
for(i=n-1;i>=0;i--)
System.out.println(a[i]);
}
catch(IOException e)
{
System.out.println(e);
}
catch(Exception e)
{
System.out.println("Error Occurred");
}
}
}
```

b)How to implement multiple inheritance? Explain with example.(2017 4b)

5. a) What is stream class and explain it’s types.

The java.io package contains a large number of stream classes that provide capabilities for processing all types data.

Input steam classes: It extracts(i.e. read) data from the source(file) and sends it to the program. Input stream classes that are used to read 8- bit bytes include a super class known as InputStream and a member of subclasses for supporting various input -related functions.

The super class inputStream is an abstract class, and therefore, we cannot create instances of this class. Rather, we must use the subclasses that inherit from this class. The InputStream class defines methods for performing input functions such as

- Reading bytes
- Closing bytes
- Marking positions in streams
- Skipping ahead in a stream
- Finding the number of bytes in a stream.

Output steam classes: It takes data from the program and sends (i.e. writes) it to the destination(file). Output stream classes are derived from the base class OutputStream . The output stream is also an abstract class and therefore we cannot instantiate it. The several sub classes of the OutputStream can be used for performing the output operations. The output Stream includes the methods that are designed to perform the following tasks:

- Writing bytes
- Closing streams
- Flushing streams

b) Briefly explain life cycle of Applet. (2017)

c) What is type casting?

We often encounter situations where there is a need to store a value of one type into a variable of another type. In such situations, we must cast the value to be stored by proceeding it with the type name in parentheses.

6. a) Write a java program to demonstrate string methods. (2017- 5B)

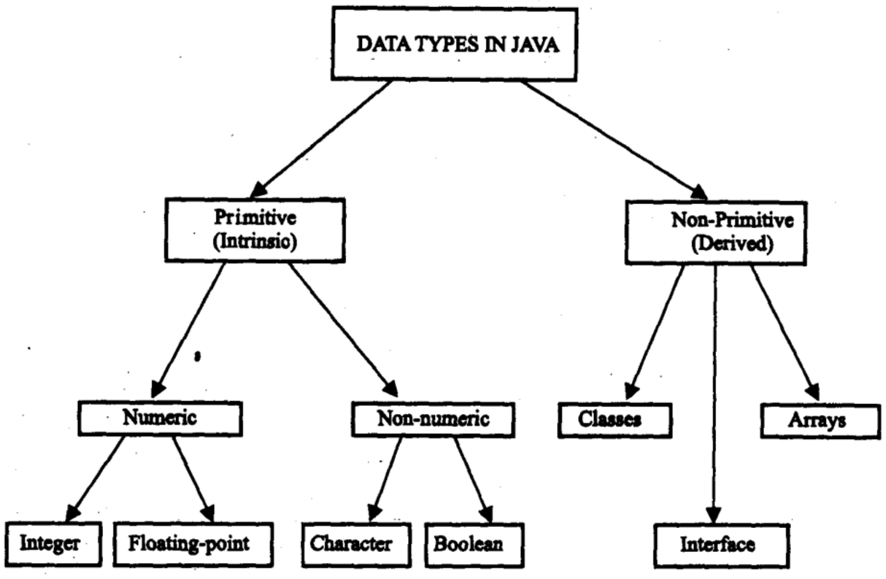
b) Briefly explain the looping statements in Java. (2017 -3b)

7. a) What are the access qualifiers? Explain with them with an example.(2017- 4C)

b) What are the data types?

Every variable in Java has a data type. Data types specify die size and type of values that can be stored. Java language is rich in its *data types.* The variety of data types available allow the programmer to select the type appropriate to the needs of the application.

Fig. 4.2



Data types in Java

Integer Types

Integer types can hold whole numbers such as 123, -96, and 5639. The size of the values that can be stored depends on the integer data type we choose. Java supports four types of integers They are byte, short, int, and long. Java does not support the concept of unsigned types and therefore all Java values are signed meaning they can be positive or negative.

Type	Size	Minimum value	Maximum value
byte	One byte	-128	127
short	Two bytes	-32,768	32, 767
Int	Four bytes	-2,147,483,648	2,147,483,647
long .	Eight bytes	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

Floating Point Types

Integer types can hold only whole numbers and therefore we use another type known as. Floating point type to hold numbers containing fractional parts such as 27.59 and -1.375 (known as floating point constants). There are two kinds of floating point storage in Java they are float and double.

Type	Size	Minimum value	Maximum value
float	4 bytes	3.4e·038	3.4e+038
double	8 bytes	1.7e-308	1.7e+308

Character Type

In order to store character constants in memory, Java provides a character data type called char.

The char type assumes a size of 2 bytes but, basically, it can hold only a single character.

Boolean Type

Boolean type is used when we want to test a particular condition during the execution of the program. There are only two values that a Boolean type can take true or false. Remember, both these words have been declared as keywords. Boolean type is denoted by the keyword Boolean and uses only one bit of storage

c) Briefly explain the Applet tag.

Note that we have included a pair of <APPLET... >and </APPLET> tags in the body section discussed above. The <APPLET ... >tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires ..The ellipsis in the tag <APPLET .. ,> indicates that it contains certain attributes that must specified. The <APPLET> tag given below specifies

```
<APPLET
CODE = helloJava.class
WIDTH = 400
HEIGHT = 200 >
</APPLET >
```

the minimum requirements to place the **HelloJava** applet on a Web page:

8. Write short notes for the following(any four).

a) Java tokens (2017 8A)

b) Vector

Vector class contained in the java.util package. This class can be used to create a generic dynamic array known as vector that can hold objects of any type and any number. The objects do not have to be homogeneous . Arrays can be easily implemented as vectors. Vectors are created like arrays as follows:

vector invect = new Vector(); //declaring without size.

Vector list = new Vector(); //declaring size

Note that a vector can be declared without specifying any size explicitly. A vector can accommodate an unknown number of items. Even, when a size is specified, this can be overlooked and a different number of items may be put into the vector. Remember, in contrast, an array must always have its size specified.

Vectors possess a number of advantages over arrays.

1. It is convenient to use vectors to store objects.
2. A vector can be used to store a list of objects that may vary in size.
3. We can add and delete objects from the list as and when required.

A major constraint in using vectors is that we cannot directly store simple data types in a vector; we can only store objects.

c) Thread priority (2017 6b)

d) Final keyword

All methods and variables to be overridden by default in subclasses. If we wish to prevent the subclasses from overriding the members of the superclass, we can declare them as final using the key word final as a modifier. Example:

final int SIZE = 100 ;
final void showstatus( ) { .... }

Making a method final ensures that the functionality defined in this method will not be altered in any way. Similarly, the value of a final variable can never be changed. Final variables, behave like class variables and they do not take any space on individual objects of the class.

FINAL CLASSES
Sometimes we may like to prevent a class being further sub classed for security reasons. A class that cannot be sub classed is called final *class*. This is achieved in Java using the keyword final as follows:
```
final class Aclass { }
final class Bclass extends Someclass { t
```
Any attempt to inherit these classes will cause an error and the compiler will not allow it. Declaring a class final prevents any unwanted extensions to the class. It also allows the compiler to perform some optimisations when a method of a final class is invoked.

e) Super

Super is a keyword of Java which refers to the immediate parent of a class and is used inside the subclass method definition for calling a method defined in the superclass. A superclass having methods as private cannot be called. Only the methods which are public and protected can be called by the keyword super. It is also used by class constructors to invoke constructors of its parent class.

Syntax:

Super<method_name>();

Uses of Super class

- Super variables refer to the variable of a variable of the parent class.
- Super() invokes the constructor of immediate parent class.
- Super refers to the method of the parent class.

f) Arrays

An array is a group of contiguous or related data items that share a common name. For instance, we can define an array name salary to represent a set of salaries of a group of employees. A particular value is indicated by writing a number called *index* number or *subscript* in brackets

after the array name. For example, salary[l0] represents the salary of the 10th employee. While the complete set of values is referred to as an *array*, the individual values are called *elements*. Arrays can be of any variable type. The ability to use a single name to represent a collection of items and to refer to an item by specifying the item number enables us to develop concise and efficient programs. For example, a loop with the subscript as the control variable can be used to read the entire array, perform calculations and, print out the results.

CREATING AN ARRAY
Like any other variables, arrays must be declared and created in. the computer memory before they are used. Creation of an array involves three steps: .
1. Declare the array
2. Create memory locations
3. Put values into the memory locations.

Creation of Arrays

After declaring an array, we need to create it in the memory. Java allows us to create arrays using new operator only, as shown below:
arrayname= new type(size);