1. List the arithmetic Operators used in Java. Write a simple Java program / Application that use basic arithmetic Operators.

Ans:- An arithmetic Operator is a mathematical function that takes two operands and performs a calculation on them. following table shows the arithmetic Operator:

Operators	Description
'+'	Add two Operands
'-'	Subtract 2nd Operands from 1st.
'*'	Multiply two Operands.
'/'	Divide numerator by denominator
% (modulus)	It find out the remainder.
++ (increment)	Increase integer value by 1.
-- (Decrement)	Decrease integer value by 1.

Java Application that use basic arithmetic Operators:

```
class BasicArithmeticOperators
{
void Calculate()
{
float a=50, b=10;
{
float add = a+b;
float sub = a-b;
float mul = a*b;
```

```
        float div = a/b;
    }
    System.out.println("Addition of a and b is:"
                        +add);
    System.out.println("subtract b from a is:"
                        +sub);
    subtract.
    System.out.println("Multiply of a and b is:"
                        +mul);
    System.out.println("Divide of a to b is:"
                        +div);
    }
    public static void main(String[] args)
    {
    BasicArithmeticOperator BAO = new BasicArithme-
        ticOperator();
    BAO.Calculate();
    }
}
```

2. Define Class and Object. Write a java program to calculate area of rectangle and display it using Concept of Class and Object.

Ans:- Class is a collection of Objects of similar type. for example Mango, Orange, Apple are the member of class fruit.

Class is basically a blueprint or a template of a set of objects which shares some common properties and behaviors. Once a class is created, we can create any number of objects belonging to that class.

(i) Class is a collection of objects.
(ii) Class is not a real world entity. It is just a blueprint or prototype.
(iii) Class does not occupy memory.

Object

Objects are the basic identifiable runtime entities in Object-Oriented programming. Object may represent a real-world entity like a person, a car, a place, a house etc. For instance, we can say 'Car' is an Object which has some special characteristics like 'number of gears', 'Colour' etc and it also holds some functions like 'breaking', 'acceleration' and so on.

```
Class AreaOfRectangle
{
void Calculate()
{
```

```
int l, b, A;
    {
    A = l*b;
    }
System.out.println("Area of rectangle is:"
                    +A);
    }
public static void main (String[] args)
    {
AreaofRectangle AOR = new AreaofRectangle();
AOR.Calculate();
    }
    }
```

3. Differentiate between Exception and Errors. Explain with example the mechanism of exception Handling in Java.

Ans:-

S.N	Exception:	S.N	Errors
(i)	The exceptions are the issues that can appear at runtime and compile time.	(i)	The error indicates trouble that primarily occurs due to the scarcity of system resources.
(ii)	It is possible to recover from an exception.	(ii)	It is not possible to recover from an error.

S.N	Exceptions	S.N	Errors
(i)	In java, the exceptions can be both checked and unchecked.	(i)	In Java, all the errors are unchecked.
(ii)	The code of the program is accountable for exceptions	(ii)	The system in which the program is running is responsible for errors.
(iii)	They are described in java.lang.Exception package.	(iii)	They are described in the java.lang.Error package.

for example

```
Class ExceptionTest
{
public static void main (String[] args)
{
try
{
int a=50, b=0;
System.out.println("1");
System.out.println("2");
System.out.println("3");
System.out.println("4");
System.out.println("5");
System.out.println(a/b);
```

```
}
catch (Exception e)
{
System.out.println(e);
}
System.out.println("6");
System.out.println("7");
System.out.println("8");
System.out.println("9");
System.out.println("10");
}
}
```

4. Write a Java program to find greatest number among 10 input numbers using array ?

Ans:-

```
import java.util.Scanner;
class A
{
public static void main(String[] args)
{
int a[] = new int[10]; int max;
Scanner s = new Scanner(System.in);
System.out.println("Enter any 10 numbers");
for(int i=0; i<10; i++)
{
```

```
        a[i] = r.nextInt();
    }
max = a[0];
for (int i=1; i<10; i++)
{
  if (a[i] > max)
  {
    max = a[i];
  }
}
System.out.println("Greatest number is :" +
    max);
}
}
```

5. Discuss the main features of Java with reference to Object Oriented programming Concepts.

Ans:- The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are known as Java buzz-words.

(i) Simple
(ii) Object-Oriented
(iii) Distributed
(iv) Robust
(v) Secure
(vi) System Independence
(vii) Portability
(viii) Interpreted
(ix) High performance
(x) Multithreaded
(xi) Dynamic

Explain Main features

(1) Simple :-

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun microsystem, Java language is a simple programming language because,

(a) Java syntax is based on C++ (so easier for programmers to learn it after C++)

(b) Java has removed many complicated and rarely-used features, for example, explicit pointers, Operator Overloading etc.

(c) There is no need to remove unreferenced objects because there is an Automatic garbage collection in Java.

(ii) Secured :-

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because, No explicit pointer.

Java programs runs inside a virtual machine sandbox.

(iii) Portable :-

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

7. How can you overload the methods in Java? Explain with an example.

Ans:- Whenever a class contain more than one method with same ~~name~~ name and different types of parameters called method overloading.

Syntax:

```
return-type method-name (para1);
return-type method-name(para, para);
```

Different ways to overload the method.

There are two ways to overload the method in Java. They are

(1) By changing number of arguments or parameters

(11) By Changing the data type.

for example

```
class A
{
    void add()
    {
    int a=2, b=5, c;
    {
    c = a+b;
    }
    system.out.println(c);
    }
    void add(int x, int y)
    {
    int c;
    {
    c = x+y;
    }
    system.out.println(c);
    }
    void add(int x, double y)
    {
    double c;
    {
    c = x+y;
    }
```

```
        System.out.println(c);
    }
    public static void main(String[] args)
    {
        A a = new A();
        a.add();
        a.add(20,25);
        a.add(50,20.5);
    }
}
```

8. Explain Interfaces with suitable Example.

Ans:- Interface is a mechanism to achieve abstraction in Java. It is similar to abstract class but having all the methods of abstract type i.e it cannot have a method body. Interfaces are the blueprint of the class. It specify what a class must do but not how. Along with abstract methods, an interface may also contain constants, default methods, static methods..

An interface is different from a class in several ways.

1. You cannot instantiate an interface.
2. An interface does not contain any constructors.
3. All of the methods in an interface are abstract.

④ An interface can extend multiple interfaces

Syntax

```
interface InterfaceName
{
Methods → public abstract type method
fields → public static final keyword
Default method
static method
}
```

Example

```
interface InterfaceName1
{
void show(); // By default public abstract void show();
}
interface InterfaceName2
{
void display();
}
Class TestMain implements InterfaceName1,
InterfaceName2
{
public void show()
{
System.out.println("This is show method");
}
public void display()
```

```
{
System.out.println("This is display method");
}
public static void main(String[] args)
{
TestMain tm = new TestMain();
tm.show();
tm.display();
}
}
```

10. Write short notes on:

(a) Encapsulation

Encapsulation is one of the most important concept of object oriented programming. It is a technique which combines both data members and functions, operate on that in a single unit known as class. This technique basically prevents the access to the data directly. The only way to access the data is provided by the function.

If you want to read a data in an object, you have to call the member function in the object. The function will read the data and return the data back to you. So, you have no access to the data

directly. Since data is hidden, it is secured from accidental alteration.

for example, Consider a big company which may have different departments like production, marketing, sales, account etc. Each department has its own manager to maintain its data. If the production manager wants to know the sales data of last month, the production department would not be allowed to himself go through the sales department data files. He will have to write a letter to the sales department requesting the data required.

(b) Method Overriding

When a method defined in the base class is redefined in the ~~base cla~~ derived class to fit its own needs, it is called method overriding. Method overriding contains a method in the derived class which has the same name and argument types as the method in the base class but the definition is different. In other words, Method overriding is the use of two or more methods having same

name with same signature but defined one in base class and other in derived class.

If derived class defines same method as defined in its base class it is known as method overriding in Java. If you create an object of the derived class and call the member method which exist in both classes (base and derived) the member method of the derived class is invoked & the method of the base class is ignored. It enables you to provide specific implementation of the method which is already provided by its base class.

Example

```
import java.util.*;
    class school
        {
  void display()
        {
system.out.println("School Method");
        }
        }
class Student extends School
        {
  void display()
        {
system.out.println("Student Method");
```

```
    }
public static void main (String [] args)
    {
School Ob1 = new School();
 Ob1. display();
Student Ob2 = new student();
   Ob2. display();
   Ob2. display();
    }
}
```

C. Static fields and methods.

Static fields

In each object of a class will have its own copy of all the fields of the class. However, in certain situations, it may be required to share a common copy of fields among all the objects of the same class. This is accomplished by declaring the field(s) to be static and such fields are known as static field(s). If a field is declared static then there is one field for the entire class instead of one per object.

Example

```
class Rectangle
{
int length; // length of rectangle
int breadth; // breadth of rectangle
static int rectCount = 0; // count rectang objects

void setData (int l, int b)
{
length = l;
breadth = b;
rectCount ++;
}
//method to calculate area of rectangle
int area()
{
int rectArea;
rectArea = length * breadth;
return rectArea;
}
}
class StaticField
{
public static void main (String [] args)
{
Rectangle firstRect = new Rectangle();
firstRect.setData(5,6);
System.out.println("Area of Rectangle 1:" +
firstRect.area());
Rectangle secondRect = new Rectangle();
secondRect.setData(10,20);
```

```
System.out.println("Area of Rectangle 2: "
        +SecondRect.area());

System.out.println("Total Number of Objects
    : " +Rectangle.rectCount);
  }
}
```

Static Method

It is possible to have static methods in a class in the same way as we have static fields. The static method is similar to instance method of a class but the only different is that the static method can be called through its class name without creating any objects of that class. A static method is also called class method as it is associated with a class and not with individual instance of the class.

Example

```
class Test
{
static void display()
{
```

```
system.out.println("It is static display
                    method");
    }
public static void main(String[] args)
    {
display(); //OK  Test.display(); //OK  show(); //
error
  xyz.show(); } }
class xyz { static void show() { system.out.println
("static show() method"); }
```

6. How does applet differ from application? What are the basic steps to create applets?

Ans:- The main difference betn. Applet and Application is that the applet is a small Java program that can be executed by a Java-compatible web browser while the application is a standalone program that can directly run on the machine.

The major difference between both of these is that an application is a standalone program that can run independently whereas an applet can run on a Java compatible web browser only. An application has access to all the resource on a system while an applet does not have any access to the resource on a system.

Applet can be achieved by following these basic steps:—

(i) Write the Java code in a text file.
(ii) Save the file.
(iii) Compile the code.
(iv) Fix any errors.
(v) Reference the applet in a HTML page.
(vi) Run the applet by viewing the web page.

9. How can you read and write to a file in Java programming? Explain with example.

Ans:- In Java, you can read and write to file using the 'Java.io' package. Here's how to do it:

Writing to a File

To write to a file, you can use the 'FileWriter' class. Here's an example:

```
import java.io.FileWriter;
import java.io.IOException;
public class WriteToFileExample {
public static void main(String[] args) {
try {
FileWriter writer = new FileWriter("example.txt");
writer.write("Hello, World!");
writer.close();
```

```
System.out.println("Succesfully wrote to the file.");
} catch (IOException e) {
System.out.println("An error occurred.");
 e.printStackTrace();
}
}
}
```

In this example, we create a 'FileWriter' object and pass the name of the file we want to write to as a parameter. we then call the 'write' method on the 'FileWriter' object to write a string to the file. Finally, we close the 'FileWriter' object.

Reading a from a File

To read from a file, you can use the 'FileReader' class. Here's an example:

```
import java.io.BufferedReader;
import Java.io.FileReader;
import Java.io.IOException;
public class ReadFromFileExample {
 public static void main (String[] args) {
  try {
 FileReader = new FileReader ("example.txt");
BufferedReader bufferedReader = new BufferReader (reader);
String line = bufferedReader.readline();
While (line != null) {
system.out.println (line);
line = bufferedReader.readline();
```

```
}
reader.close();
} catch (IOException e) {
System.out.println("An error occurred");
e.printStackTrace();
}
}
}
```

In this example, we create a 'FileReader' object and pass the name of the file we want to read from as a parameter. We then create a 'BufferedReader' object and pass the 'FileReader' object to its constructor. We use the 'readLine' method of the 'BufferedReader' object to read each line of the file, and print it to the console. Finally, we close the 'FileReader' object.