CS3391 - Object Oriented Programming.

Unit-1 INTRODUCTION TO OOP AND JAVA FUNDAMENTAL

Overview of OOP - object Oriented programming paradigms - Features of object oriented programming - Java Buzzwords - Overview of Java - Data types, Variables and arrays - operators - control statements - programming structures in java - Defining classes in Java constructors - Methods - Access specifiers - static members - Java Doc comments.

Overview of OOPS:

In object oriented programming approach there is a collection of objects.

Classification:

1. classes and objects
2. Abstraction
3. Encapsulation
4. Inheritance
5. Polymorphism

OOP Paradigm :

Classes :

Class can be defined as an entity in which data and function are put together.

Syntax :

```
class name_of_class
{
  private :
      variable declaration;
      function declaration;
  public :
      variable declaration;
      function declaration;
};
```

Objects :

Object is an instance of class.

Syntax :

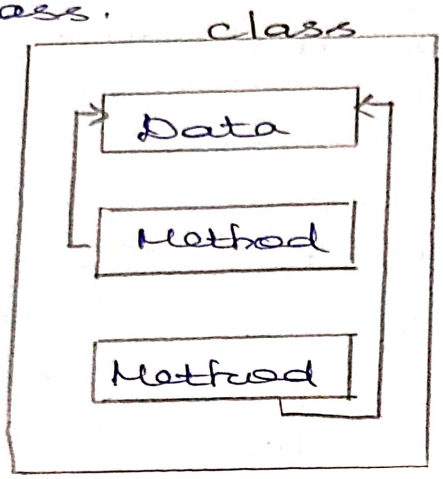
```
class name object_Name;
```

Abstraction :

Abstraction means representing only essential features by hiding all the implementation details.

Encapsulation:

Encapsulation means binding of data and method together in a single entity called class.



Inheritance:

Inheritance is the property by which new classes are created from old classes

Types:

1. Single inheritance
2. Multilevel inheritance
3. Multiple inheritance
4. Hierarchical inheritance
5. Hybrid inheritance

Polymorphism:

Polymorphism is the ability to take more than one form and refers to an operation exhibiting different behaviour in different instances.

Characteristics of Java:

Java is getting popular because of its features. Following is a list of buzzwords that make Java popular.

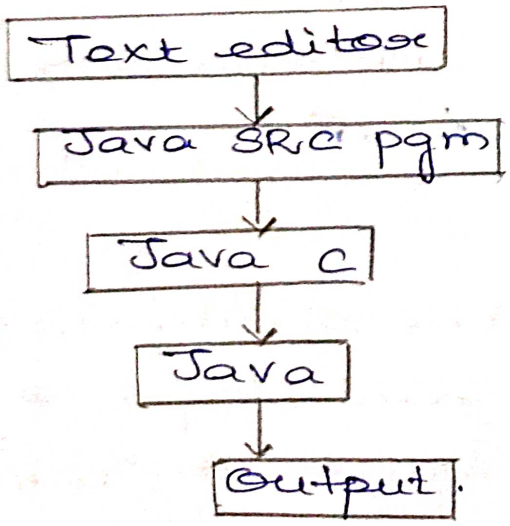
Simple
Secure
Platform independent
Portable
Robust

Overview of Java:

JDK - Java Development kit

Java development kit is a collection of tools used for development and run time progress.

Execution Process:



Data types:

1. int (2 or 4 bytes)
2. long (8 byte)
3. float (4 byte)
4. char (1 byte)
5. double (8 byte)
6. Boolean (True or false)

Variable:

Variable is an identifier that denotes storage location.

Syntax:

```
datatype name_of_variable;
```

Array:

Array is a collection of similar type of elements

Syntax:

```
datatype array_name[];
```

Types:

1. One - dimensional array
2. Two - dimensional array

One dimensional array:

ex:

```
class Sample
{
 public static void main (String args[])
{
 int a[] = new int [3];
 system.out.println ("Storing array:");
```

```
        a[0] = 1;
        a[1] = 2;
        a[2] = 3;
        System.out.println("Element at a[2] = "+a[
    }
}
```

Two dimensional array:

Two dimensional array are the arrays in which elements are stored in rows and columns.

Syntax:

datatype - array-name = new datatype[size

Operators:

Operators are the symbol used in the expression for evaluating them.

Types:

1. Arithmetic operator
2. Relational operator
3. Logical operator
4. Assignment operator
5. Increment operator
6. Decrement operator

Arithmetic Operator:

Arithmetic operator are used to perform basic arithmetic operations

+, -, *, /, % are the operators.

ex:

```
class Arithmetic
{
 public static void main (String args[])
{
 int a=10, b=20, c;
  c=a+b;
 System.out.println ("Addition ="+c);
 }
}
```

Relational operators:

Relational operators establish the relation among the operators. The result is in Boolean value.

<, >, <=, >= are relational operators.

ex:

```
class Relational
{
 public static void main(String args[])
 {
 int a=10, b=20, c;
 if (a>b)
 {
  System.out.println("a is big");
 }
 else
 {
 System.out.println("b is big");
 }
```

Logical operator:

Logical operators are used to co, ce two operands these two are of boolean operators.

ex:

```
class Logical
{
  public static void main (String args[])
  {
    boolean op1 = true, op2 = false;
    boolean ans1, ans2;
    ans1 = op1 & op2;
    ans2 = op1 | op2;
    System.out.println (+ans1);
    System.out.println (+ans2);
  }
}
```

Assignment Operator:

Assignment operator are used to assign a value to a variable. '=' equal to is an assignment operator.

Conditional operator:

The conditional operator is "?".

Syntax:

Condition ? statement 1: Statement 2

Control statements:

Programmers can take decision in their programs with the help of control statement.

i) if statement
ii) if-else
iii) while
iv) Do-while
v) Switch
vi) for loop

If statement:

Types:

1. Simple
2. Compound

Syntax for simple:

```
if (condition)
statement;
```

Syntax for compound:

```
if (condition)
{
  statement 1;
  statement 2;
}
```

ex:

```
class demo
{
  public static void main (String args[])
  {
   int a=10, b=5;
   if (a>b)
  System.out.println ("a is big");
  }
}
```

If-else statement:

syntax:

```
if (condition)
   statement;
else
   statement;
```

ex:

```
class demo
{
 public static void main (String args[])
 {
 int a=2, b=5;
 if(a>b)
  System.out.println ("a is big");
  else
  System.out.println ("b is big");
 }
}
```

if-else-if :

Syntax :

```
if (condition)
  statement;
elseif (condition 2)
statement;
else
  statement;
```

ex :

```
class demo
{
public static void main (String args[])
 {
  int a=5, b=5;
  if (a>b)
 System.out.println("a is big");
 else if (b>a)
 System.out.println("b is big");
 else
   System.out.println("Both are equal");
 }
}
```

while statement:

Syntax:

```
while (condition)
{
  statement 1;
  :
  statement n;
}
```

ex:

```
class demo
{
 public static void main (String args[])
 {
  int count=1;
  int i=0;
  while (count<=2)
  {
    i=i+1;
    System.out.println (+i);
    count++;
  }
 }
}
```

do-while statement:

Syntax:

```
do
{
  statement 1;
  :
  statement n;
}while (condition);
```

ex:

```
class Sample
{
 public static void main (String [] args)
{
int count =1, i = 0;
do
 {
  i = i+1;
  System.out.println ("+i);
  count++;
 }while (count <=3)
}
}
```

For loop:

Syntax:

```
for (initialization ; condition ; inc/dec)
 {
  statement 1;
    :
  statement n;
}
```

ex:

```
class Sample
{
 public static void main (String args[])
{
 for (int i=0; i<=3; i++)
```

Switch case:

```
class Sample
{
 public static void main(String args[])
 throws java.io.IO.Exception
 {
  int choice;
  System.out.println("1.A");
  System.out.println("2.B");
  System.out.println("Enter choice");
  choice=(int)System.in.read();
  switch(choice)
  {
  case1:
   System.out.println("You Selected A");
   break;
  case2:
   System.out.println("You selected B");
   break;
  default:
     System.out.println("Nore");
   }
 }
}
```

Programming Structures in Java.

Java tokens:

The smallest and logical, individual unit of the java statements are called tokens.

Types:

1. Reserved words
2. Identifiers
3. Literals
4. Operators
5. Separators.

Reserved words:

Keywords are reserved words.

int, float, char are keywords.

Identifiers:

Identifiers are defined by users. They are used for naming class, object, Variables etc.

Literals:

Literals are used to store the sequence of characters for representing the constant values.

Types:

1. Integer
2. Floating Point
3. Boolean

4. Character
5. String

Constructors:

Syntax:

```
class Test
{
   Test ()
   {
   }
}
```

ex:

```
class Customer
{
  int km;
  int m;
  Distance (int k, int m)
  {
    km=k;
     meter=m;
  }
}
```

```
Public static void main (String args[])
{
 Distance shop = new Distance (1, 200);
 Distance hotel = new Distance (2, 300);
}
```

Access Specifiers:

Access modifiers control access to data fields, methods and classes.

Types:

1. Public
2. Private
3. Default modifier

Public:

Allows classes, methods and data fields accessible from any class.

Private:

Allows classes, methods and data field accessible from only within the own class

ex:

```
Package Test
Public class class1
{
  public int a;
  int b;
  private int c;
  public void fun1()
   {
   }
  void fun2()
  {
  }
Private void fun3()
```

```
    }
    }
}
Public class class2
{
    void my-method()
    {
    class1 obj = new class1();
    obj.a;
   obj.b;
    obj.c;
    obj.fun1();
    obj.fun2();
    obj.fun3();
  }
}
Package architex Test
public class class2
{
  void my meth()
  {
  class1 obj = new class1();
  obj.a;
  obj.b;
  obj.c;
  obj.fun1();
  obj.fun2();
  }
}
```

Java Doc Comments:

Javadoc is a convenient, standard way to document your java code. Javadoc is actually a special format of comments

Types:

1. class level comments
2. Member level comments

Methods:

ex:

```
void get_data (int a, int b)
{
int c = a+b;
}
public class Method
{
public static void main (String args[])
{
Method m= new Method;
m sum (10,20);
}
void sum (int a, int b)
{
  int c;
  c = a+b;
  System.out.println (c);
}
}
```

Static members:-

Static members are those members which can be accessed without using objec

example:

```
Class ex
{
  Static int a=10;
  {
   System.out.println("a="+a);
  }
}

class anotherclass
{
  public static void main (String [] args)
  {
   System.out.println("a="+ex.a);
  }
}
```

o/p

a=10