Control Statements, Classes, Objects & Methods

1. Control statements in java

1.1 If statements in java

'If' is a decision making statement in java. It will execute the block of code when the condition is true; if condition fails it skips the execution.

Syntax:

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

Program:

```
public class Greater {
    public static void main(String[] args) {
        int x = 20, int y=30;
        if (x > y)
            System.out.println("x > y");
        if (x < y)
            System.out.println("y > x");
    }
}
```

Output:

Y> x

Program: 2

```
public class Add{
    public static void main(String[] args) {
int x=4
```

```
        if (x == 4)
            System.out.println("x is equal to 4");
        If(x != 4)
System.out.println ("x is not equal to 4");
    }
}
```

Output:

X is equal to 4

1.2If else statement:

In some cases we have to perform some action if our condition is not satisfied in those cases 'If else' is used. 'If else' statement is similar to 'if' statement, apart from the appended 'else' statement. 'If else' statement follows the Boolean expression. If the statement is true, statement will be executed otherwise 'else' block will be executed.

Syntax:

```
If (condition) {
  //Executes when the condition is true
}
 else
{
  //Executes when the condition is false
}
```

Program:

```
public class Add{
    public static void main(String[] args) {
        int x=4
        if (x == 4)
```

```
            System.out.println("x is equal to 4");
        else
            System.out.println ("x is not equal to 4");
    }
}
```

Output:

x is equal to 4

1.3 while loop:

While loop will execute the statement if the condition is true. It will execute the single statement or block of code with in loop:

Syntax:

```
while(condition){
        statement
}
```

Program:

```
public class SakshIEducation {
        public static void main (String[] args) {
                int count = 1;
                System.out.println("Displaying Numbers from 1 to 10");
                while (count <= 10) {
                        System.out.println(count++);
                }
        }
}
```

Output:

Displaying Numbers from 1 to 10

1

2

3

4

5

6

7

8

9

10

1.4 do while loop

The ‘do-while’ loop is similar to the ‘while’ loop, except in the case that, in do-while test is performed at the end of the loop, while in the ‘while’ test is performed at the beginning. This comes handy in the cases when we want the loop to go with the first iteration and based on the first iteration result we have to decide whether to continue or not. The ‘do-while’ loop statement will be executed once, do while loop starts with the keyword ‘do’

Syntax:

```
do
{
  //Statement
}
while (condition);
```

Program:

```
public class SakshiEducation {

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

```
        int count = 1;
        System.out.println("Displaying Numbers from 1 to 10");
        do {
            System.out.println(count++);
        } while (count <= 10);
    }
    }
```

Output:

Displaying Numbers from 1 to 10

1

2

3

4

5

6

7

8

9

10

1.5 'for' loop:

"**for**" loop executes the group of statements until the condition is true. The notable difference between while and for is: in the former case the only the condition statement is given, but where as in the later case we can specify the initialization condition and increment within the same statement.

Syntax:

```
for (initialization, condition, increment)
{
        statements;
}
```

1. Initialization is used to initialize the expression (int a=1)
2. The second one is condition (a>=20). as long as condition is true loop will execute.
3. The third is the modifying expression used to increment or decrement the variable

Program:

```
public class ForLoopDemo {


        public static void main(String args[]) {
                System.out.println("Displaying Numbers from 1 to 10");
                for (int count = 1; count <= 10; count++) {
                        System.out.println(count);
                }
        }
}
```

Output

```
Displaying Numbers from 1 to 10
1
2
3
```

4

5

6

7

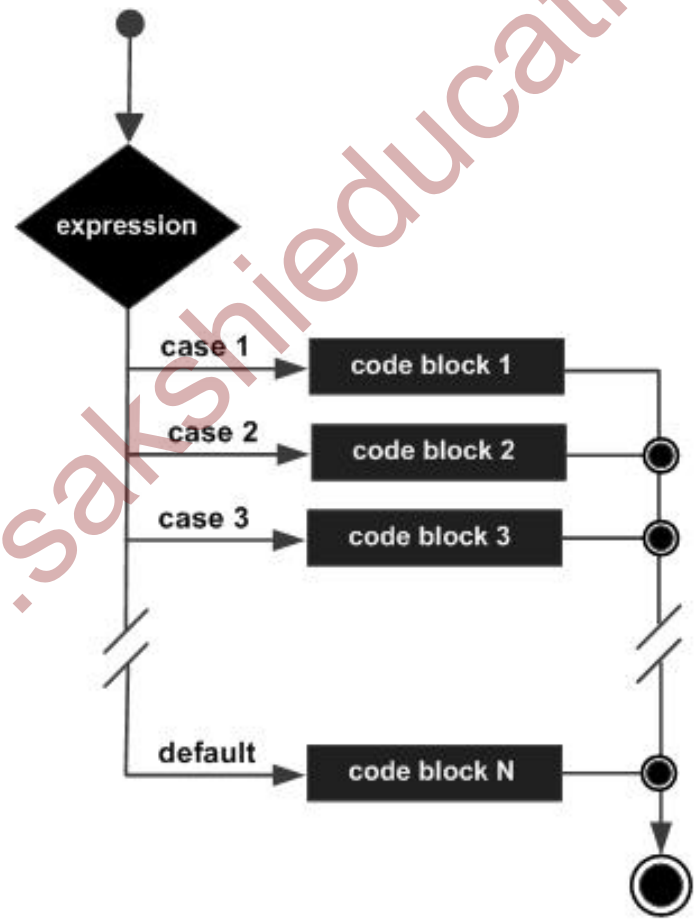
8

9

10

1.6 'switch' statement:

Switch statement is used to select one option among the several others, based on that condition statements mentioned in the switch statement.



Source: http://www.tutorialspoint.com/

Syntax:

```
switch(expression) {
   case value1:
        statement1
      break;
   case value2:
        statement 2
      break;
   default:
default statement will be executed
}
```

Program:

```
String name = "Sakshi";
switch (name) {
case "Sakshieducation”:
    System.out.println("This is Sakshieducation.");
    break;
case "Sakshi" :
    System.out.println("This is sakshi.");
    break;
case "Sakshipost" :
    System.out.println("This is sakshipost.");
    break;
default:
    System.out.println("Not a Valid Name.");
```

}

Output: This is Sakshi

1.7 break statement:

Break statement is used to terminate from the loop. Break statement is used in for loop, while loop, do while loop, if we use break statement inside the loop, immediately it will execute the statement after the loop. Break comes handy in handling infinite loops and recursive loops.

Syntax: break;

Break statement in for loop:

```
for(initialization, condition, increment){
        statement1;
        statement 2;
        statement 3;
        break;
        statement 4;
}
Statement n;
```

Note: in the above program after break directly **statement n** will be executed

1.8 continue statement

At times program demands to continue run the loop by halting the reminding code for specific iterationl, in these cases Continue statement comes handy. This skips, the part of the code and execute the next statement. This statement causes control to be transferred directly to the conditional expression that controls the loop.

Syntax:

```
{
Statement 1
Statement 2
continue;
```

statement 3

statement 4

}

3 Concept of classes & objects & constructors & methods

3.1 class

The impelling concern over data security has led to introduction of Concept 'class' in Object oriented programming language, which are pivoted towards the functional approach, where some specific set of functions and data are enveloped in to a Unit called 'Class' and functions are the only means to access the data.

A class is nothing but a blueprint or a template for creating different objects which defines its properties and behaviors. Java class objects exhibit the properties and behaviors defined by its class. A class can contain fields and methods to describe the behavior of an object.

Program:

```
public class Animal{
  String breed;
  int age;
  String color;
  void barking(){
  }
  void hungry(){
  }
  void sleeping(){
  }
}
```

Program-2:

```
public class Cube {
```

```
        int length;
        int breadth;
        int height;
        public int getVolume() {
                return (length * breadth * height);
        }
}
```

3.2 Objects:

The term object refers to an actual instance of a class. Objects facilitate us to use access the data and functions in the class. Every object must belong to a class. Objects are created and eventually destroyed – so they only live in the program for a limited time. While objects are living their properties may also be changed significantly.

You can represent real-world objects using software objects. You might want to represent real-world dogs as software objects in an animation program or a real-world bicycle as a software object within an electronic exercise bike. However, you can also use software objects to model abstract concepts.

3 steps to create an object from a class:

Declaration:

A variable declaration with a variable name with an object type.

Instantiation:

The 'new' key word is used to create the object.

Initialization:

The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Program:

```
public class Sakshi{
  public Sakshi(String name){
    // This constructor has one parameter, name.
    System.out.println("Name is :" + name );
  }
  public static void main(String []args){
    // Following statement would create an object sakshiEducation
    Sakshi sakshiEducation = new Sakshi (“sakshipost”);
  }
}
```

Output:

Name is : sakshipost

3.3 constructors

Constructor is a special type of method that is used to initialize the object.

Note:

- Constructor name must be same as its class name
- Constructor must have no explicit return type

There are two types of constructors:

1. Default constructor
2. Parameterized constructor

3.3.1 Default constructor:

constructor that have no parameter is known as default constructor

program:

```
class Apple{
Apple(){
System.out.println("Apple is a Fruit");
}
public static void main(String args[]){
Applefruit a=new Applefruit();
}
}
```

Output: Apple is a Fruit

3.3.2 Parameterized constructor:

A constructor with parameters is known as parameterized constructor.

Program:

```
class Animal{
   int id;
   String name;
   Animal(int i,String n){
   id = i;
   name = n;
   }
   void display(){System.out.println(id+" "+name);}
   public static void main(String args[]){
   Animal x1 = new Animal (00,"Monkey");
   Animal x2 = new Animal (11,"Zebra");
   x1.display();
```

```
    x2.display();
  }
}
```

Output:

00 Monkey

11 Zebra

In above program, we have created the constructor of Animal class that have two parameters. We can have any number of parameters in the constructor

3.4 methods:

A methods is a collection of statements grouped together to perform an operation.

Syntax:

```
Return type methodName(parameters)
{
        //method body
}
```

Example:

```
public String getName(String sr)
{
Stiring Name= "Sakshi education";
name name+sr;
return name;
}
```

public : modifier is a access type of method

return type: a method may return value. Data type of value by a method declared in a method haeding

method name: name of the method

parameters: parameters passed to the method

method body; statements that defined in method

3.5 'this' keyword

'this' is a keyword in Java, which can be used inside method or constructor of class. It works as a reference to current object whose method or constructor is being invoked. 'this' keyword can be used to refer any member of current object from within an instance method or a constructor.

Program:

```
class Animal{
   int id;
   String name;
   Animal(int i,String n){
   this.id = id;
   this.name = name;     }
   void display(){System.out.println(id+" "+name);}
   public static void main(String args[]){
   Animal x1 = new Animal (00,"Monkey");
   Animal x2 = new Animal (11,"Zebra");
   x1.display();
   x2.display();
  } }
```

Output:

00 Monkey

11 Zebra

Test

1. What is the base class of all classes?

   Ans: java.lang.Object

2. Are arrays primitive data types?
   Ans: In Java, Arrays are objects

3. Is Java a pure object oriented language?
   Ans: Java uses primitive data types and hence is not a pure object oriented language
4. When will you define a method as static?

   Ans: When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.
5. What is constructor?
   Ans: Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation

6. What is the purpose of default constructor?
   Ans: The default constructor provides the default values to the objects. The java compiler creates a default constructor only if there is no constructor in the class

7. Is constructor inherited?
   Ans: No, constructor is not inherited

8. Can you use this() and super() both in a constructor?
   Ans: No. Because super() or this() must be the first statement.

9. Can we override the overloaded method?
   Ans: Yes.

10. Can we execute a program without main() method?
    Ans: Yes, one of the way is static block