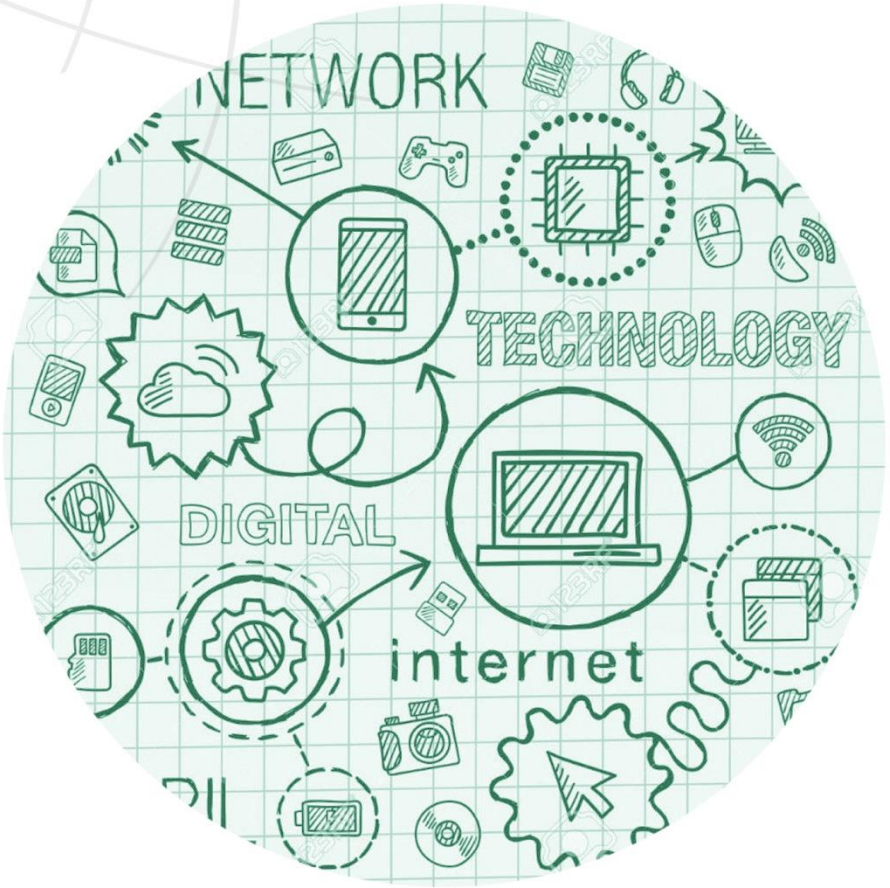
myperfectice

COMPUTER SCIENCE AND ENGINEERING



Quick Guide

On

Java Concepts

www.myperfectice.com

Java Concepts

Java - Overriding

- We have talked about super classes and subclasses. If a class inherits a method from its superclass, then there is a chance to override the method provided that it is not marked final.
- The benefit of overriding is: ability to define a behavior that's specific to the subclass type, which means a subclass can implement a parent class method based on its requirement.

Example

```
class Animal {
  public void move() {
    System.out.println("Animals can move");
  }
}
class Dog extends Animal {
  public void move() {
    System.out.println("Dogs can walk and run");
  }
}
public class TestDog {
  public static void main(String args[]) {
    Animal a = new Animal();   // Animal reference and object
    Animal b = new Dog();   // Animal reference but Dog object
    a.move();   // runs the method in Animal class
    b.move();   // runs the method in Dog class
  }
}
```

This will produce the following result:

Output

```
Animals can move
Dogs can walk and run
```

Rules for Method Overriding

- The argument list should be exactly the same as that of the overridden method.
- The return type should be the same or a subtype of the return type declared in the original overridden method in the superclass.
- A method declared final cannot be overridden.
- A method declared static cannot be overridden but can be re-declared.
- If a method cannot be inherited, then it cannot be overridden.
- Constructors cannot be overridden.

Using the super Keyword

When invoking a superclass version of an overridden method the super keyword is used.

Example

```
class Animal {
  public void move() {
    System.out.println("Animals can move");
  }
}
class Dog extends Animal {
```

```
public void move() {
    super.move();   // invokes the super class method
    System.out.println("Dogs can walk and run");
  }
}
public class TestDog {
  public static void main(String args[]) {
    Animal b = new Dog();   // Animal reference but Dog object
    b.move();   // runs the method in Dog class
  }
}
```

This will produce the following result:

Output

```
Animals can move
Dogs can walk and run
```

Java - Methods

A Java method is a collection of statements that are grouped together to perform an operation.

Creating Method

Considering the following example to explain the syntax of a method −

Syntax

```
public static int method Name(int a, int b) {
  // body
}
```

Here,

- **public static** − modifier
- **int** − return type
- **methodName** − name of the method
- **a, b** − formal parameters
- **int a, int b** − list of parameters

Method definition consists of a method header and a method body. The same is shown in the following syntax −

Syntax

```
modifier returnType nameOfMethod (Parameter List) {
  // method body
}
```

Example

Here is the source code of the above defined method called min(). This method takes two parameters num1 and num2 and returns the maximum between the two −

```
/** the snippet returns the minimum between two numbers */
public static int min Function(int n1, int n2) {
  int min;
  if (n1 > n2)
    min = n2;
  else
    min = n1;

  return min;
}
```

The void Keyword
The void keyword allows us to create methods which do not return a value.

The this keyword
This is a keyword in Java which is used as a reference to the object of the current class, with in an instance method or a constructor. Using *this* you can refer the members of a class such as constructors, variables and methods.

Variable Arguments(var-args)
- JDK 1.5 enables you to pass a variable number of arguments of the same type to a method. The parameter in the method is declared as follows −
- Type Name... parameter Name
- In the method declaration, you specify the type followed by an ellipsis (...). Only one variable-length parameter may be specified in a method, and this parameter must be the last parameter. Any regular parameters must precede it.

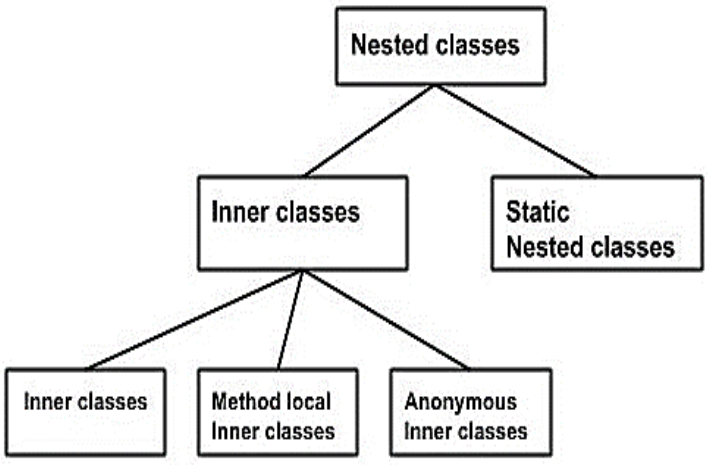
The finalize( ) Method
It is possible to define a method that will be called just before an object's final destruction by the garbage collector. This method is called **finalize( )**, and it can be used to ensure that an object terminates cleanly.

Java - Inner classes
Nested Classes
In Java, just like methods, variables of a class too can have another class as its member. Writing a class within another is allowed in Java. The class written within is called the **nested class**, and the class that holds the inner class is called the **outer class**.

Syntax
Following is the syntax to write a nested class. Here, the class Outer_Demo is the outer class and the class Inner_Demo is the nested class.

```
class Outer_Demo {
  class Inner_Demo {
  }
}
```

Nested classes are divided into two types:
- **Non-static nested classes** − These are the non-static members of a class.
- **Static nested classes** − These are the static members of a class.



Inner Class

Creating an inner class is quite simple. You just need to write a class within a class. Unlike a class, an inner class can be private and once you declare an inner class private, it cannot be accessed from an object outside the class.

Method-local Inner Class

- In Java, we can write a class within a method and this will be a local type. Like local variables, the scope of the inner class is restricted within the method.
- A method-local inner class can be instantiated only within the method where the inner class is defined. The following program shows how to use a method-local inner class.

Anonymous Inner Class

An inner class declared without a class name is known as an **anonymous inner class**.

Static Nested Class

A static inner class is a nested class which is a static member of the outer class. It can be accessed without instantiating the outer class, using other static members. Just like static members, a static nested class does not have access to the instance variables and methods of the outer class. The syntax of static nested class is as follows –

Syntax

```
class My Outer {
  static class Nested_Demo {
  }
}
```

MyPerfectice Points:

- When a class inherits another class, it can override the method of base class. Overriding means that the derived class can provide an implementation of the same method that is already provided by the base class.
- Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by **super** reference variable.
- **This** is a keyword in Java which is used as a reference to the object of the current class, with in an instance method or a constructor
- Finalize() method is a method that is invoked just before an object's final destruction by the garbage collector.

We bring best of techology, analysis, teaching and mentoring to prepare you smartly.
You reach your goal in shortest distance, time and money.

myperfectice

Practice. Analyze. Improve.

hello@myperfectice.com
www.myperfectice.com