Java Concepts :- Part -03

So far we had covered the basic of Java programming, which will help you to understand some basic properties of Java and that how does Java language works. Now we will take a deeper dive in Java and will see some of the useful and important features of Java programming language.

Some of the important properties and features mentioned below.

1. Static scope rule

* Instance variable :- Instance variable are declared in a class, but outside a method, constructor or any look.

When space is allocated for an object in the heap, a slot for each instance variable is created.

The instance variable are visible for all methods, constructors, and block in the class.

* Class variable :- Class variable also known as static variable declared with the static keyword in a class, but outside a method, constructor or a block. There would only be one copy of each class variable per class, regardless of how many objects are created from it.

Example :-

Public class VariableExample{

Int myVariable;

Static int data = 30;

Public static void main(String args[]) {

Int a = 100;

variableExample obj = new VariableExample();

System.out.println(“Value of instance variable myVariable: ” + obj.myVariable);

System.out.println(“Value of static variable data: ” + VariableExample.data);

}

}

Output :- Value of instance variable myVariable: 0

Value of static variable data: 30

1. Static :-

* Java does not have global variable.
* Every variable in Java be declared inside a class.
* The keyword “static” is used to make a variable just like global variable.
* A variable declared with static keyword is called “class variable.”
* It acts like a global variable, that is, there is only one copy of the variable associated with the class. That is, one copy of the variable regardless of the number of instance of the class.

1. Nested class :- Nested class means a class inside another class. We can define as many nested class as much we want. But we should try not to declare many nested class because it makes the code confusing and chances of getting error increase.
2. Recursion in Java :- Recursion in Java is a process in which a method is calls itself continuously. A method in Java that calls itself is called recursive method.

It makes the code compact but complex to understand.

Example :-

Public class RecursionExample2 {

Static int count = 0;

Static void p(){

Count ++;

If(count <= 5){

System.out.println(“Hello”+ count);

P();

}

}

Public static void main(String args[]) {

P();

}

}

1. While loop v/s do-while loop :- while loop checks the condition first and if the condition is true than only it runs, while on other hand do-while loop run for the first time and after that it checks the condition.
2. Inheritance in Java :-

* Inheritance is one of the cornerstone of object-oriented programming because it allows the creation of hierarchical classification.
* Using Inheritance, one can create a general class that includes some common set of items.
* This class then can be used to create more specific classes which has all the items from the base class, in addition to some items of it’s own.

1. Terms used in inheritance :-

* Superclass :- A class that is inherited is called a superclass.
* Subclass :- The class that does inheriting is called a subclass.

A subclass is a specialized version of a superclass.

It inherits all of the instance variables and methods defined by the superclass and add it’s own, unique elements (i.e. variable and methods)

* Reusability :- It is a mechanism which facilitates you to reuse the data and methods of the existing class when one create a new class.

One can use the same data and methods already defined in the previous class.

1. Inheritance syntax :- The “extend” keyword is used to define a new class that derives from an existing class. The meaning of “extends” is to increase the functionality.

Class <Subclass-name> extends <Superclass-name> {

// data and methods in this sub-class

}

1. Inheritance types :-

* Single Inheritance
* Multiple single inheritance
* Multilevel single inheritance

1. Method overriding concept :-

* Usage of Java method overriding :- Method overriding is used to provide the specific implementation of a method is already provided by it’s superclass.

Method overriding is used for runtime polymorphism.

* + Rules for Java Overriding :- The method must have the same name as in the parent class.
    - The method must have the same parameter as in the parent class.
    - There must be an IS-A relationship (inheritance).

Note :- A sub class object can reference a super class variable or method if it is not overridden.

* A super class object cannot reference a variable or method which is explicit to the sub class object.

1. Super keyword concept in Java :-

* The “super” keyword in Java is a reference variable which is used to refer immediate parent class members.
* Whenever you create an instance of a sub class, an “instance of it’s parent class is created implicitly”, which is referred by “super” keyword.

Example :-

class Animal {

String color = "white";

}

class Dog extends Animal {

String color = "black";

void printColor() {

System.out.println(color);

System.out.println(super.color);

}

}

class TestSuper {

public static void main(String args[]) {

Dog d = new Dog();

d.printColor();

}

}

Output :- black

White

Animal and Dog classes have a common property colour. “If you print colour property, it will print the colour of the current class by default ”. To accessthe parent property, you should use “super” keyword.

Example :-

class Animal{

Animal() {

System.out.println("animal is created");

}

}

class Dog extends Animal {

Dog() {

super();

System.out.println("dog is created");

}

}

class TestSuper3 {

public static void main(String args[]) {

Dog d = new Dog();

}

}

Output : -animal is created

Dog is created

The “super” keyword can also be used to invoke the overloaded parent class constructor, if arguments are there, then they should be specified accordingly

1. Dynamic method dispatch concept :
   * Dynamic method dispatch is a process in which a call to an overridden method “is resolved at runtime” rather than compile-time. Also, it is called “Runtime polymorphism.”
   * In this process, an overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable.
2. Abstract concept :-

* “Abstraction” is a process of hiding the implementation details and showing only functionality to the user.
* Abstraction let’s you focus on what the object does instead of how it does it.
* A class which is declared with the “abstract” keyword is known as an “abstract class” in Java. It can have abstract and non-abstract methods (i.e. method with the body only without its definition.)
* Points to remember :- 1.) An “abstract class” must be declared with an “abstract” keyword.

2.) It can have abstract and non-abstract “methods”.

3.) It “cannot” be instantiated.

4.) It can have constructors and static methods also.

5.) It can have “final methods” which will force the sub class not to change the body of the method.

Example :-

abstract class Bike {

abstract void run();

}

class Honda extends Bike {

void run() {

System.out.println("Running safely");

}

public static void main(String args[]) {

Bike obj = new Honda();

obj.run();

}

}

Output :- Running safely

Here, “Bike” is an “abstract class” that contains only one abstract method run(). It implementation is provided by the “Honda” class.

Note :- An abstract method should be defined in it sub class.

1. Final keyword concept :-

* The “final” keyword in Java is used to restrict the access of an item from its super class to a sub class. The Java “final” keyword can be used in many contexts.
* Variable :- A variable cannot be accessed in sub class.
* Method :- A method cannot called from a sub class object.
* Class :- A class cannot be sub classed.

Note :- If you make any class as “final”, you cannot extend it.

1. Concept of access modifiers :- The access modifiers in Java specify accessibility (scope) of a data member, methods, constructors or class.

This is the third part of the article series for “Java Concepts”, if you find it helpful than make sure to read the next article also. If you want to refresh the basic concepts you can always refer to previous post, and it is also recommended.

If you have any doubt, question or query related this topic, then please feel free to contact me.