Quick Revisit

JAVA – High Level, Platform Independent Programming Language

JAVA is Case Sensitive.

1. Learn all the character-set, alphabets, operators available in the language
2. Keywords, statements, expression, variables
3. Classes & Objects, OOP concepts
4. Exception Handling, Collection, Thread

Objects will be created in Heap area. Class is the blue-print

Objects – Instance of the class

Class – Is the blue-print to create object.

class Student {

}

class Trainer {

}

All the keywords should be in lower case only.

Keywords in Java - <https://www.geeksforgeeks.org/list-of-all-java-keywords/>

Student <obj\_ref\_name> = new Student();

To create an Object, we use the class name and after the assignment operator we call the constructor along with the new keyword.

Day9 – Project Folder

---- src (This is folder where all java source code will be saved - .java)

---- com

----revature

Student.java

---- bin (This is the folder where all java byte code will be saved - .class)

---- com

----revature

Student.class

To create an Object we need class.

Static members are once per class. Non-static members are once per object.

Static members needs to be accessed using the class name, where as non-static members needs to be accessed using the object name.

Abstract – in-complete/non-concrete –

Abstract methods are in-complete/non-concrete methods which have declaration only not the definition.

Public abstract void show() ; -- This is abstract method

Pillers of OOP (Object Oriented Programming)

1. Abstraction – Hiding the implementation
2. Inheritance - Way one class acquire members of other class, re-use existing code
3. Polymorphism – Way of re-using the code. ( instead of re-inventing the wheel, using existing to build on top of it)
4. Encapsulation – hiding the data (Capsuling the data) – It’s a way of securing the data.

Inheritance – This is a way of acquiring the property of another class (Parent – Child relationship)

Parent – The one who is giving the property – base / super class

Child – The one who is receiving the property – Derived / Sub class

Inheritance is achieved with the help of “extends” keyword

Types of Inheritance

1. Simple inheritance (parent -> child)
2. Multiple inheritance (parent1 & parent2 -🡪 child)
3. Multi-level inheritance (parent 🡪 child 🡪 grand child or grand parent -🡪parent 🡪 child)
4. Hybrid inheritance (Multiple & multi-level )

Field/property/instance variable/ state – all are same

In Interface all the members are public.

In interface we no need to specify abstract keyword for abstract methods.

A class can extend only one other class

Multiple – inheritance is not allowed in java for classes.

Multiple – inheritance is allowed in java for interfaces.

Class child extends parent { } -- This is valid (simple inheritance)

Class child extends parent1,parent2 { } – This is invalid (Multiple inheritance)

Interface a extedns b,c { } – This is valid in java (Multiple inheritance for interfaces)

Abstraction – Showing relevant data and hiding irrelevant data (maps) –

Car dashboard hides the complex electrical wiring to Head lights, indicators, horns etc.,

In Java, abstraction is achieved with the help of abstract keyword and interfaces

In Interface we can have n number of abstract methods.

Also we can have static and default concrete methods (Method with body)

Polymorphism – Poly -Many morphism – (Forms) (One method but Different forms or many forms)

1. Static polymorphism (method overloading) – Re-using the same method with different signature [changing arguments data types, changing number of arguments ) – compile time polymorphism
2. Dynamic polymorphism (method overriding)

Super() – calling the base class constructor

@Override – Just inform the JVM about there is a method defined in its parent class, the same method is re-defined in the child class

1. Overloading – within the same class using same method for different purpose
2. Overriding – Between parent & child class, redefining the same method which is already defined in its parent class.

super – refers to parent/base

this – refers to current object

super() – refers to parent class constructor

this() – refers to current object

static methods can’t be overridden.

Encapsulation – It’s a way of Securing the data.

Java is called as a highly secured language, bcos the data is not available to external world.

Class is the combination of state & behavior. Here in java, the data is secured by declaring it as a private and only accessed by the public members of the same class.

Getters & Setters, Constructor – These three methods are very important in terms of accessing the private data of the class.

Constructor is used when the object gets created. To create an object, we are calling either the default/parameterized constructor.

Constructor is a special method, which will have same name as the class name.

Constructor will not return anything and there won’t be any return type for constructor method.

Constructor can be overloaded. We can have n number constructors in a class.

Main purpose of constructor method is to initialize all the member variables to it’s default value.

If the variable is primitive, it will initialize with zero (byte, int, float, long, double =0, boolean=false, object = null, char =’\0’)

After creating the object, if you want to modify it’s properties, then setters method will be useful.

Encapsulation is achieved by making the member variables private and member methods as public. With the help of public methods we can access the private members.

Getters – Accessors

Setters – Mutators

Type Casting – Is the process of converting one data type to another data type

1. Implicit /Automatic Type casting – No additional keywords required – Widening
2. Explicit /Manual Type casting – Needs to add casting keywords – Narrowing (Down casting)
3. Boolean🡪byte🡪short🡪char 🡪int🡪float🡪double🡪long (smaller to bigger)
4. Boolean 🡨 byte 🡨 short 🡨 char 🡨 int 🡨 float 🡨 double 🡨 long (bigger to smaller)

String

1. In JAVA, String is a Special Type.
2. String class is defined in java.lang package
3. It supports both primitive & object declaration
4. In Java, String are immutable in nature (It can’t be modified after the creation)
5. When ever we do some operation on String, it will re-create another String object.
6. To avoid memory related issues when performing more String related operations, we can use either StringBuffer or StringBuilder class
7. Both StringBuffer & StringBuilder are mutable String classes. It have append() & insert() method
8. When calling substring method, the begin index is inclusive and end index is exclusive.

Object (java.lang)

1. In Java, everything is Object.
2. Unless specified for all the user defined classes , java.lang.Object is the default super class.
3. Every User-defined class will get methods of Object class (equals(), hashcode(), etc.,)

Abstract Class –

1. It’s a in-complete/non – concrete class
2. If a class has any abstract methods, then the class needs to be declared as a abstract class.
3. For an Abstract class, objects can’t be created directly.

MyAbstract obj = **new** MyAbstract() {

@Override

**public** **void** display() {

// **TODO** Auto-generated method stub

System.***out***.println("Display method!!!");

}

};

Class NO\_NAME {

@Override

**public** **void** display() {

// **TODO** Auto-generated method stub

System.***out***.println("Display method!!!");

}

}

For defining a contract use interface.

1. A class can extend only one class ,
2. A class can implement n number of interfaces

Interfaces – this is used to define contract between the code & developer

Interface provide specification. Implementation class will implement those specification.

Every day - 1 to 2 hours for self reading. Revisit all the concepts discussed on that day.

In case of doubts, queries you can write an email to me.