Topics

* Core Java
* JDBC, Servlets & JSP
* HTML, CSS, Typescript, Javascript
* Git
* UML
* Junit
* Spring Framework
* Spring Boot & Microservices
* Python
* Cloud & Containers
* Design Patterns
* React.js

Java

It is a platform independent & object oriented programming language.

Platform Independent: You can run java programs on various platforms without re-compiling

Object Oriented Language: You create real world entities in the application and define the properties & behaviors of these real world entities, these real world entities are called as objects.

Two things are must in object oriented language which are also called as building blocks of an object oriented language

1. class: blueprint or template of an object
2. object: an instance created from the class

run -> entry point -> C/C++/Java -> main method

Java: public static void main(String[] args) { }

class Hello {   
 public static void main(String[] args) {   
 System.out.println(….);  
 }  
}

String[] - it is a command line argument which accepts the input from the user before launching the main

static - you can access static members without creating the object

public - visible outside the class

Softwares required

1. JDK - javac, java commands
2. Editor - notepad, vscode or IDE like Eclipse

Eclipse -> File -> Other -> Java Project

Fundamentals

1. Variables - these are the memory to store the data
2. Datatypes - byte, short, int, long, float, double, char, boolean, arrays & complex types
3. Operators - \*, -, +, /, ++, --, <=, >=, ==, >, <, %, !=, &&, ||
4. Conditional statements - if else .. if else if .. else, switch
5. Loops - for, while, do while
6. Jump statements - break, continue, return

Java Naming conventions

avoid using variable names like a, b, c, i, j, k, x, y, z as they don’t explain themselves what they are instead use variable names like name, age, phoneNumber, gender and so on

Similarly for methods / functions you must use the names which are having meanings like

display(), update(), getDetails(), searchEmployee(), searchCustomer() and so on, avoid using test(), demo(), abc() and so on.

While or Do While loop

While loop: It first checks the condition & then executes the loop

Do while loop: at least once the loop is executed and then the condition is checked to continue

for each or enhanced for loop iterates the elements without using the index.

Activity:

Create an array of numbers like 1, 4, 3, 6, 2, 5, 0, -5, iterate over an array and display the maximum, minimum and sum of elements in the array  
ex: Max: 6, Min: -5, Sum : 16

Note: Use only one loop to display max, min & sum

For the same array search an element 2 that must print 2 is found  
then search an element 9 that must print 9 is not found

Command line arguments: These are the inputs which you can provide while running the program so that programs can use these inputs before starting, the String[] in the main is a command line argument you can pass while using java command or in eclipse you can pass through Run As option.

i.e.,

java TestApp 20 30 40   
java TestApp Hello Everyone

Type conversion: Converting one type to another type, it can be automatic or explicit depending on the type of variable you are using

Auto-widening (Type promotion)  
int x = 25; // int = 4, long = 8  
long y = x; // type promotion

Explicit-narrowing

long x = 25; // long = 8, int = 4  
int y = (int)x; // explicit narrowing

Auto-boxing & Auto-unboxing: These are automatic because it deals with converting primitives to wrappers and vice versa

int x = 25; //  
Integer y = x; // Integer = int >> Auto-boxing, 25 will stored as an Integer object

int z = y; // int = Integer >> Auto-unboxing

class Calculator {   
void add(byte x, byte y) {   
  
 }  
}  
int a = 25;  
int b = 30;  
add((byte)a, (byte)b);

Object Oriented Programming

class: It defines the structure, it can have following members

* variables or fields
* methods: Will have reusable logics
* constructors: are called when objects are created, you can have initialization logics

object: It is an instance of a class which is created with new keyword

User user1 = new User();

Constructor vs Methods

|  |  |
| --- | --- |
| Constructor | Method |
| Constructors are called when objects are created | Methods are not called when objects are created |
| It will not have return type | It must have return type |
| Its name must be same as the class name | It’s name need not to be same as the class name, however you can give class name but they are invoked only when you call them explicitly |
| It usually will have initialization logic | It will have reusable logics |

Activity:

Create User class with id, name, constructor to initialize the id & name, display method to print id & name, in main method create 2 objects & call the display method.

Classes can have static and non-static(instance) members

static members you can access directly from the class name, however non-static members you can access only by creating the object whenever you want these members to accessed outside the class.

class User {   
 int id;

static int counter = 5;  
}  
User.counter = 7; // valid  
User.id = 33; // invalid, because id is instance variable

static members are loaded at the time class loading, non-static members are loaded at the time of object creation.



Quiz:

int x = 20;

int y = ++x;

System.out.println(y); //21  
System.out.println(x); // 21

1. class A {   
    static void demo() { test(); } // error  
    void test() { System.out.println(“hello”); }  
   }
2. class A {  
    static void demo() { A a = new A(); a.test(); } // ok  
    void test() { System.out.println(“hello”); }  
   }

}

class A {   
 // default constructor in case you have not written  
}   
class B {   
 B(int x) {   
 }  
 // only one constructor will be created which takes int parameter  
}

byte x = 30;  
long y = 40;  
int z = (int) y; //   
int z1 = x; //

Principles of OOPs

Inheritance  
Encapsulation  
Polymorphism  
Abstraction

Inheritance:

Acquiring the properties & behaviors of a class from another class, you use extends keyword to inherit   
class Person {   
 name, gender, email, dob // properties  
 display() { } // display data   
}  
class Employee extends Person {   
 employeeId, salary   
}  
class Student extends Person {   
 rollNo, grade  
}

Note:

* In Java you can achieve various types of inheritance like single level, multi-level, hierarchical, multiple(supported only through interface) & hybrid (supported only through interface)
* If a class doesn’t extend any class it automatically extends Object class
* Subclass constructors calls the default constructors of the parent class by default, however you can pass parameters to the super() to call the parameterized constructor of the parent class

Polymorphism:

A method that has many forms or a method that can do more than one job, there are two types of polymorphism

1. Compile time – overloading – method invocation is determined at compile
2. Runtime – overriding – method invocation is determined at runtime

class Person, class Employee extends Person, class Student extends Person { }   
  
Method Overriding: you will create the same method of super class in the subclass with different logic & to achieve runtime polymorphism you will use the super class reference variable and call the overridden method  
void test(long x) { }   
int a = 20; test(a);  
long b = 30; test(b);  
byte c = 40; test(c);

Method Overloading: You will create multiple methods with the same name in the same class with different signature(parameter type or order or return type)

Encapsulation: It is mechanism of hiding the data by making data private & the only way you can access the data is using public methods, private data will not be visible outside the class so that it restricts the direct access from the outsider

class Employee {   
 private int id;  
 private int age;  
 Employee(int id) { this.id = id; }  
 public void setAge(int age) { this.age = age; }  
 public int getAge() { return age; }  
 public int getId() { return id; }  
}  
Employee emp = new Employee(34);  
emp.id = 356; // error  
emp.age = 4939; // error