Core Java

Software Requirement

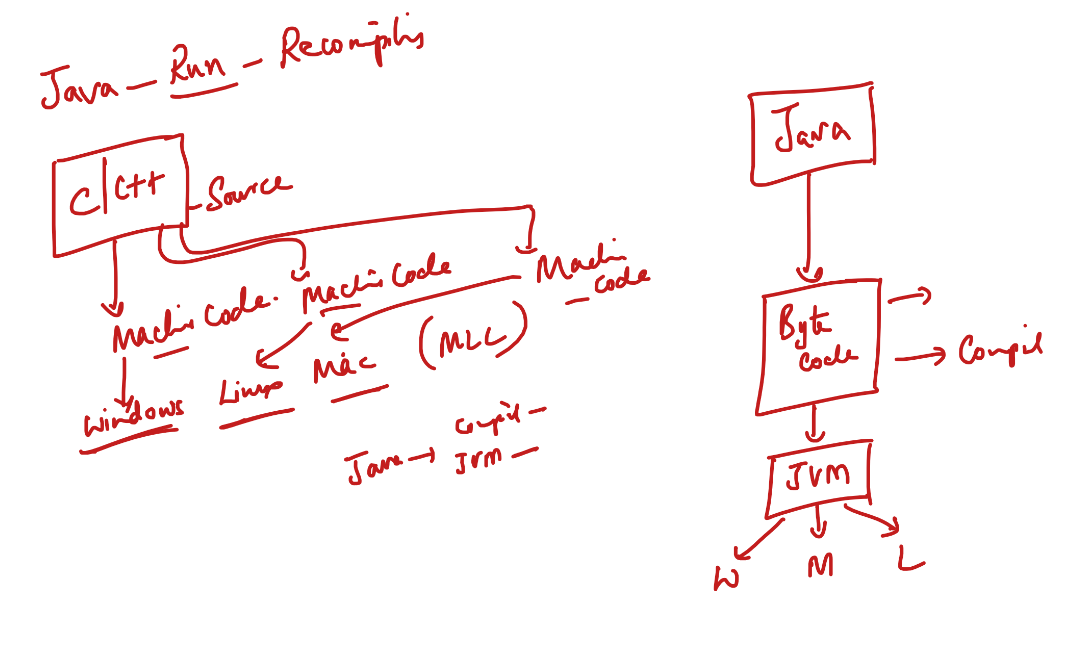
* JDK 17 or later
* Eclipse IDE / VS Code

How to verify java installation

Two commands

1. javac
2. java

Java: It is a Platform Independent & Object oriented programming language



Object Oriented Programming Language

It helps programmers to create real world entities in the application and make these real world entities to interact with another real world entities to complete any task.

Building blocks of any object oriented programming language

1. class: Blueprint of an object, that represents object structure
2. object: It is an instance of a class / real world entity that you can create from the class

Terms in Java

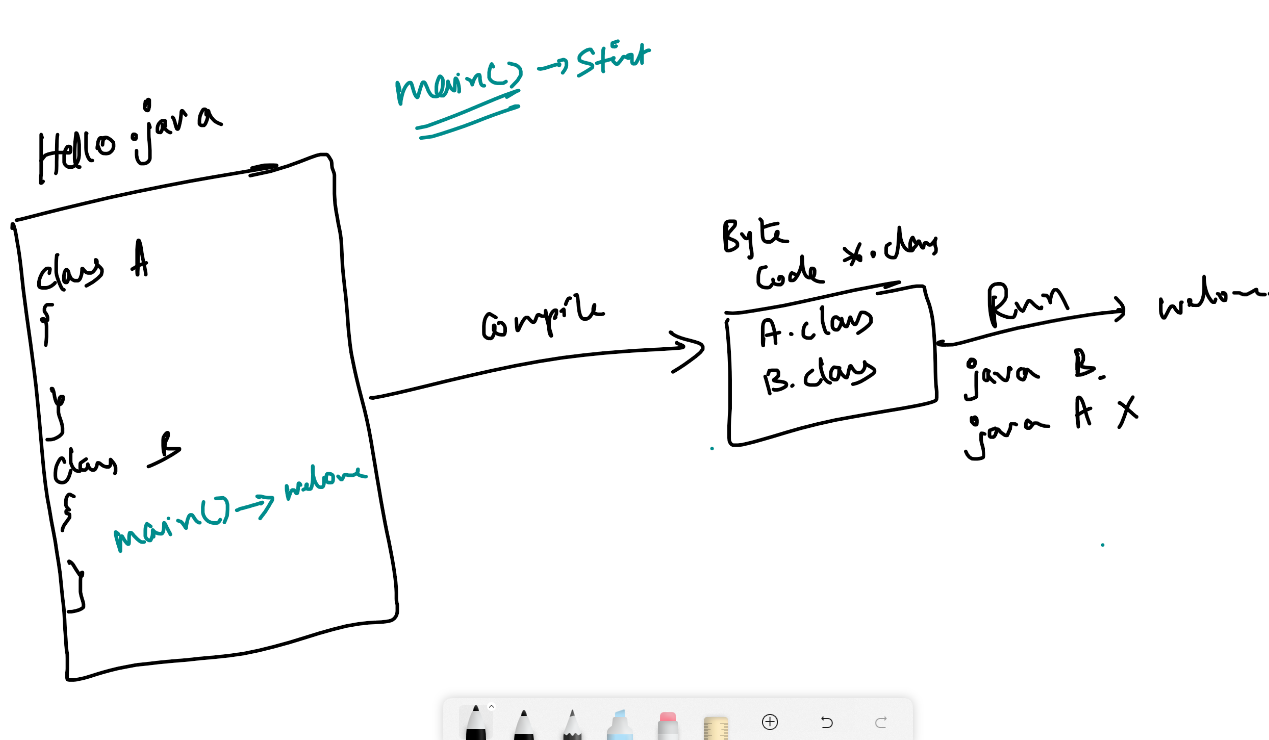
1. JDK: Java Development Kit – software that gives compiler and JRE both
2. JRE: Java Runtime Environment – place where your java application runs, it will have JVM and other components to load the application
3. JVM: Java Virtual Machine – an interpreter to convert byte code to machine code

Java versions: Java 8, Java 11, Java 17, Java 21

Java Editions:

1. JSE: Java Standard Edition – where you can create java programs in console based application
2. JEE : Java Enterprise Edition – where you can create java programs in web based applications

How to write our first java program



Fundamentals

1. Datatypes
2. Operators
3. Conditional Statements
4. Loops
5. Arrays
6. Classes & Objects

Datatypes: These are keywords used to create variables that specifies what kind of value it stores

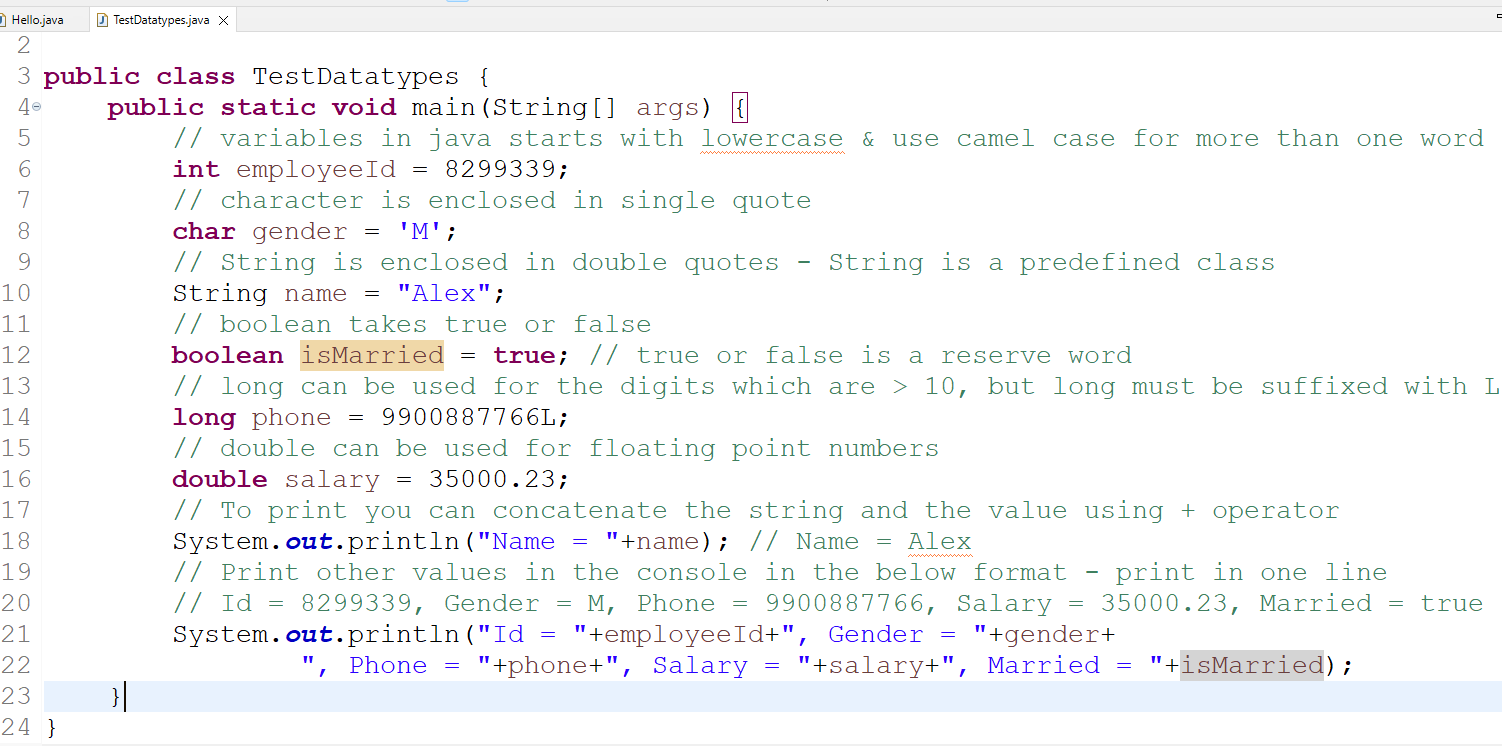
int x = 20;

double y = 55.4;

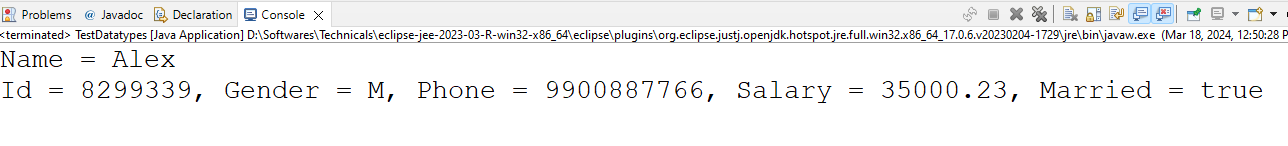
Java has following datatypes & each datatypes has fixed size

|  |  |  |
| --- | --- | --- |
| Datatype | Size in bytes | Range |
| byte | 1 | -128 to +127, including 0 |
| short | 2 | -32768 to +32767, with 0 |
| int | 4 |  |
| long | 8 |  |
| float | 4 |  |
| double | 8 |  |
| char | 2 |  |
| boolean | 1 |  |

TestDatatypes.java



Output:



Same program which takes input from the keyboard, instead of hard coding the values

In Java we have an inbuilt class Scanner in java.util package, we must import this

import java.util.Scanner;

Then we must create a Scanner object to take input from the keyboard

Scanner scan = new Scanner(System.in);

You must call inbuilt methods of Scanner to take different types of value

int x = scan.nextInt(); // to read int value  
double y = scan.nextDouble(); // to read double value  
String z = scan.next(); // to read a single string word  
boolean b = scan.nextBoolean(); // to read a boolean  
long l = scan.nextLong(); // to read a long

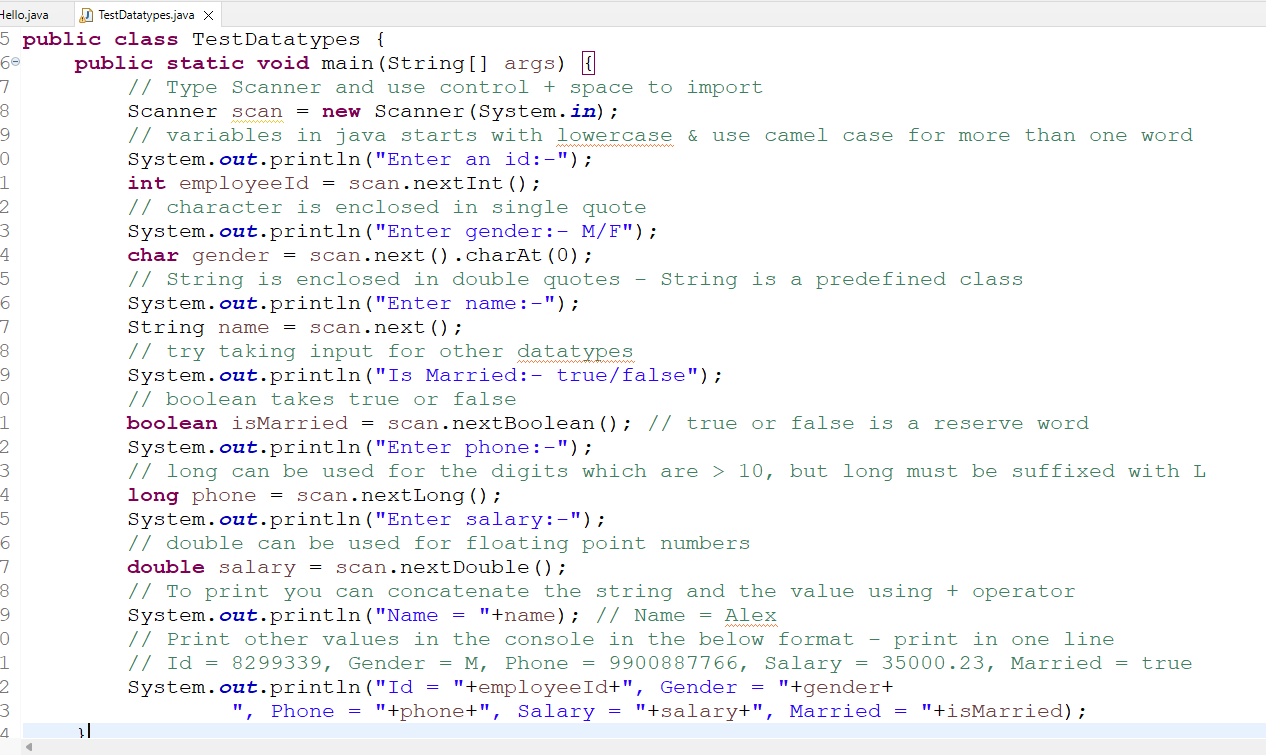
To read character you don’t have a method in scanner, you must use String and call the charAt(index) method to get a character of a specific string value

char ch = “Hello”.charAt(1); // ch = e

String z = scan.next();  
char ch = z.charAt(0); // very first character

char ch = scan.next().charAt(0); // ch = very first character of the string

TestDatatypes.java



Operators:

=, +, -, \*, /, <, >, <=, >=, ==, ++, --

x = 5;

y = x++; // post increment

Conditional Statements

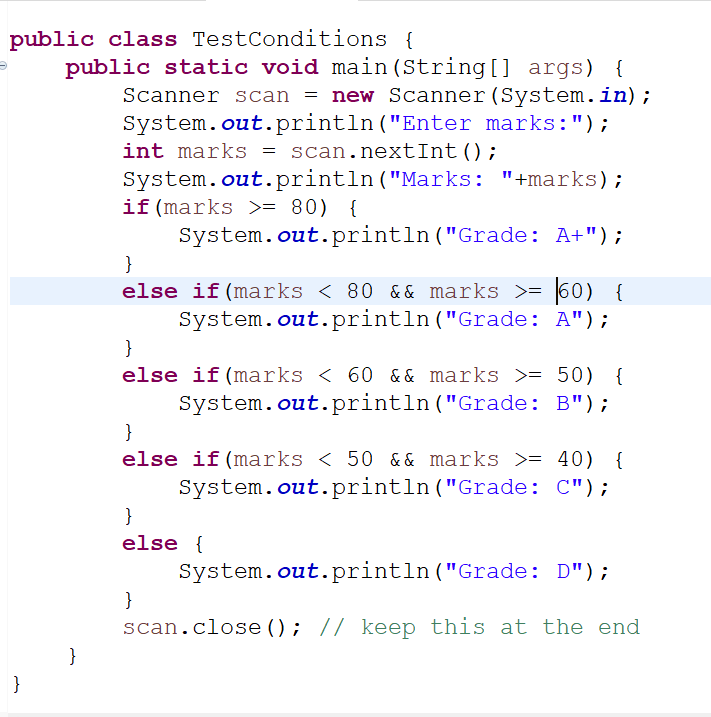
if

if else

if else if else if else

switch

TestConditions.java



TestConditions.java with switch

switch(options) {  
 case optionValue: statements;  
 break;  
 case optionValue: statements;  
 break;  
 default : statements  
}

Here optionValues can be int, String, char  
case 1 or case “someString” or case ‘A’

Loops

1. for
2. while
3. do while

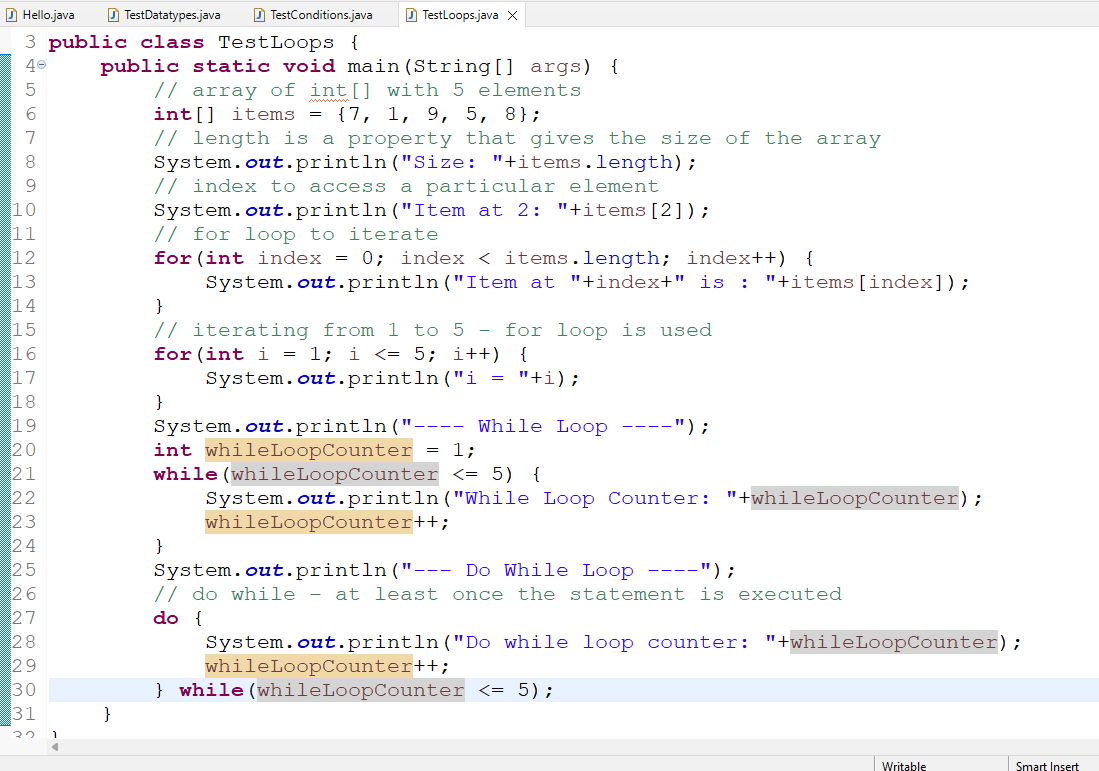
for loop: When you know how many times you want to iterate

while & do while: When you don’t know how many times you want to iterate but you want to iterate until some condition true

while: first condition is checked & then executed, its like checking the condition in the beginning

do while: At least once the statement executed and then the condition is checked, its like checking the condition while exiting

TestLoops.java



Activity:

1. Create an array of numbers and print the maximum and minimum value in the array & the sum of all the elements in the array using only one for loop

ex: items = {7, 1, -7, 8, 9, 3}, Max = 9, Min = -7, Sum = 21

1. Enter a 3 digit number and print each digit in words

ex: Input = 762, Output = Seven, Six, Two

1. Enter a 3 digit number & add the highest and lowest digit of the 3 digits

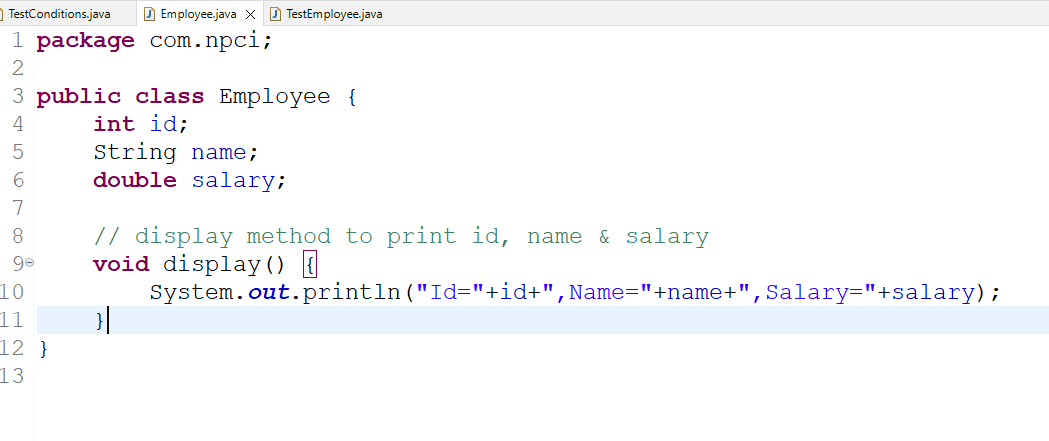
ex: Input = 762, Output = 7 + 2 = 9

Classes & Objects

class can have three things mainly

1. variables/fields/properties: Stores data
2. methods/functions: Performs action, it will have logics
3. constructors: Its like a methods but its sole job is to initialize the properties

Employee.java



TestEmployee.java



Constructors: These are called when object is created, it will be created by default if the class doesn’t have any constructor, such constructors are called as default constructors which doesn’t take any parameter, however you can overload the constructor by passing parameters

class Employee {   
 Employee(int id, String name, double salary) { //parameterized constructor }  
}

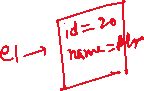
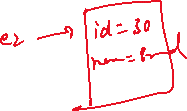
class Employee has only one constructor, default constructor wouldn’t be created.

class Employee {   
 Employee() { //default constructor }  
 Employee(int id, String name, double salary) { //parameterized constructor }  
}

Now class Employee has 2 constructors

Parameterized constructor can be used to initialize the properties

class Employee {   
 int id;  
 String name;  
 Employee(int id, String name) {   
 this.id = id;  
 this.name = name;  
 }  
}



Employee e1 = new Employee(20, “Alex”);  
Employee e2 = new Employee(30, “Brad”);

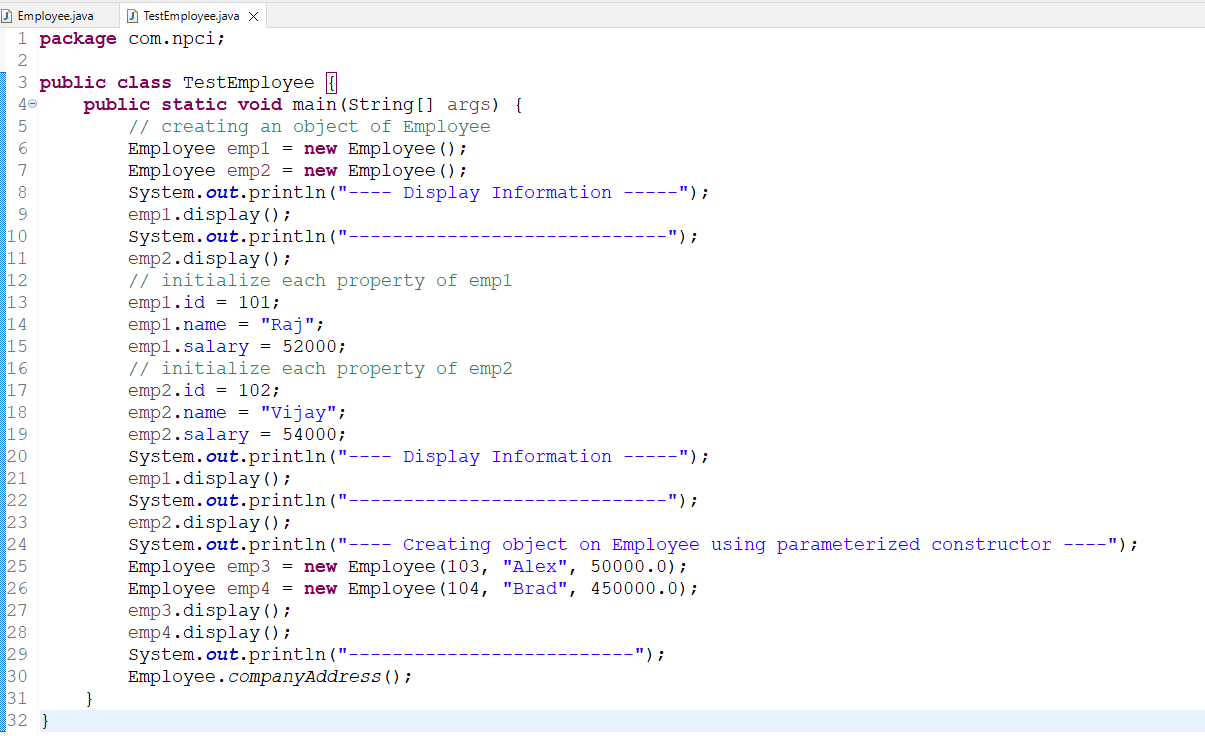
Static members:

These are the members which you can access without creating object, i.e., using the class name, you can create static variable & static methods in a class that you can call directly using the class name.

Employee.java



TestEmployee.java



Note:

1. Static methods cannot directly access non-static members(variables or methods) of the same class, it must be accessed using the object reference
2. Non-static methods can directly access static members of the same class

Final: Can be used on variables, methods & classes

final variables are constants you can’t modify

i.e., final int x = 20; // you can’t change the value of x any more

final methods & final classes we will discuss after the inheritance

class Employee {   
 final static String companyName = “NPCI”; // constant  
 static int employeeCounter = 0; // not a constant you can modify  
 Employee(…) {  
 companyName = “NPCI”; // compile time error  
 employeeCounter++;  
 }  
}

OOPs principles

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction

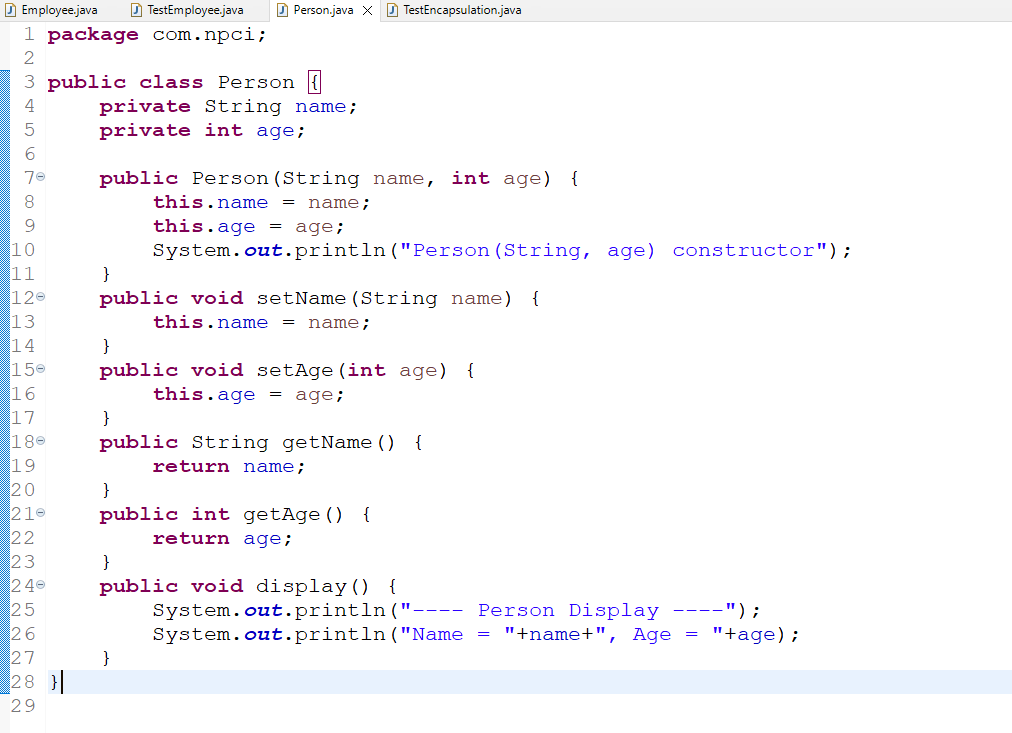
Encapsulation: It is used to hide the data so that it can’t be accessed directly outside the class, i.e., the data is restricted from the direct access to avoid code to give some invalid value

In Encapsulation you need to access this data using public setters & getters methods

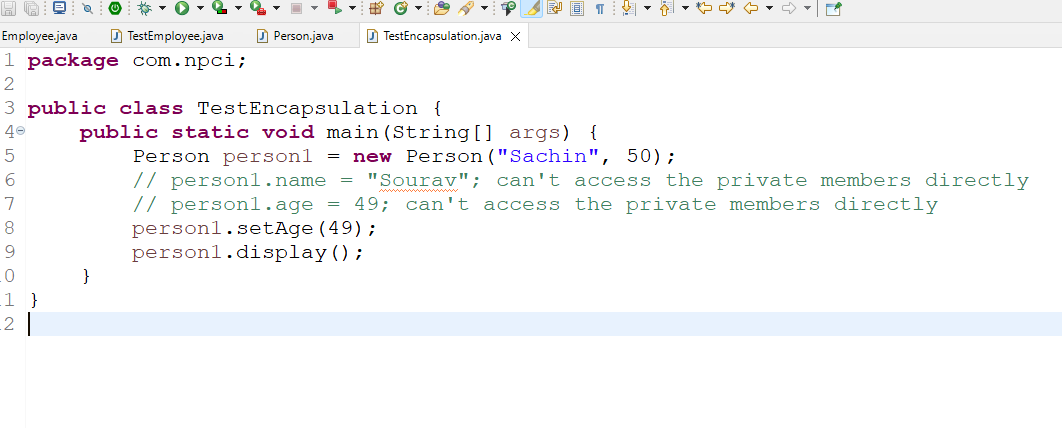
setter method: This modifies the data

getter method: This returns the data

Person.java



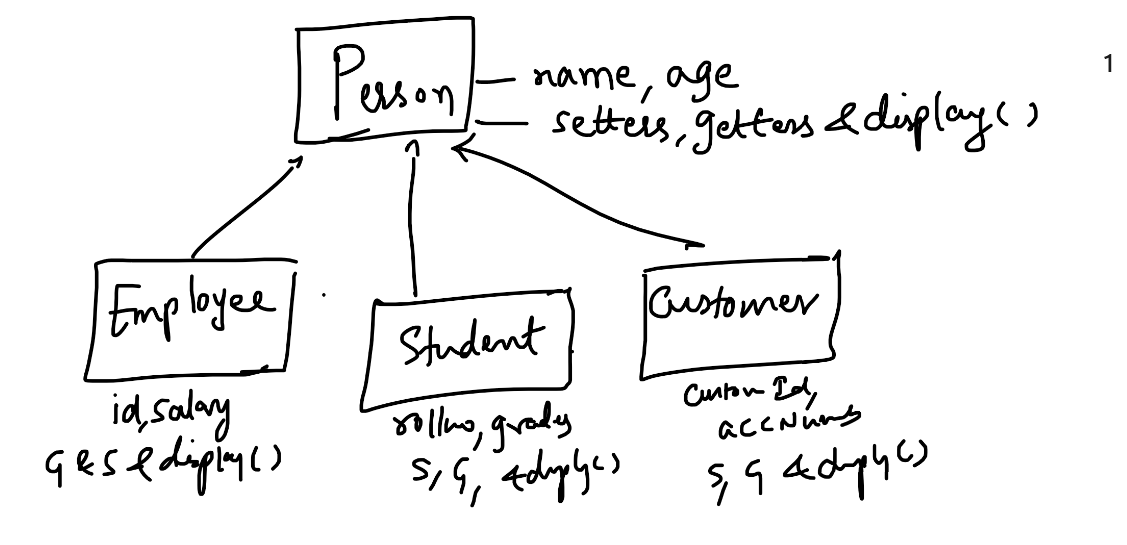
TestEncapculation.java



Note: In this example we have made the properties private so that outside the class they can’t be accessed directly, the only way is to access them using getters & setters

Inheritance

It acquires common properties & behaviors of a class from another class



Note:

1. Every constructor of the subclass by default calls the parent class default constructor
2. super() is the statement used to call the constructor, this must be present only in the first line of the constructor, keeping anywhere else leads to an error.

Object: It is the root class for all the classes, it has some common features which will be inherited to every class

Activity:

Create a Student class with rollNo, name, age and array of marks properties, initialize the name & age from the super class & override display method to print student properties, display method must calculate average marks and print the average marks

ex: if marks array has 6 marks then display prints the average value of 6 marks

In the main class create a Student object by passing rollNo, name, age and array of marks having 6 subjects & call the display method

Polymorphism: A method that has multiple forms or can give multiple results

1. method overloading: same method names but different signature, compiler can predict which method will be called in method overloading
2. method overriding: same method & same signature, but different logics in the subclass, only at runtime the method call can be predicted, here a single statement can give multiple results

System.out.println(2);

System.out.println(“Hello World”);

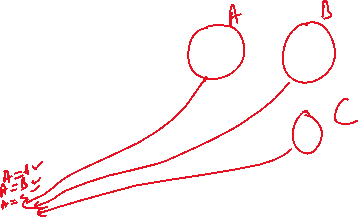
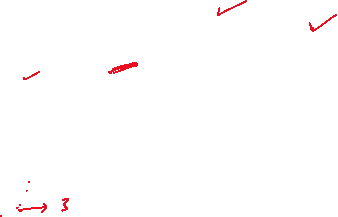
System.out.println(x); // println can take all types of data like boolean, int, long, String, Object

class Calculator {  
 void add(int x, int y) { }   
 void add(double x, double y) { }   
}

add(20, 30); // add(int, int)  
add(20.3, 40.3); // add(double, double)

Method overriding

class A {   
 m1() { … }   
}  
class B extends A {   
 m1() { … }  
}  
class C extends B {   
 m1() { … }  
}



void test(A x) {   
 x.m1();  
}



Abstraction

Hiding the complexity and showing only the necessary details to the end user, this makes code loosely coupled

It is achieved in 2 ways

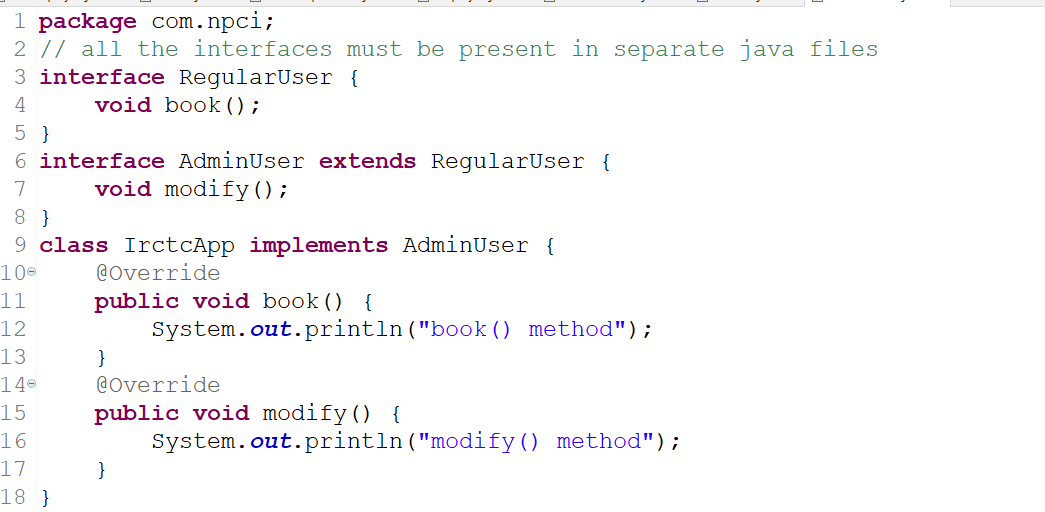
1. interface
2. abstract class

interface: It is a contract that specifies what methods do instead how it is implemented, it will have only method declarations (methods without logic)

interface X {   
 void m1(); // public abstract void m1();  
 void m2(); // public abstract void m2()  
}

class One implements X {   
 // class must implement all the abstract methods of the interface  
}  
class Two implements X {   
 // class must implement all the abstract methods of the interface  
}

TestInterfaces.java

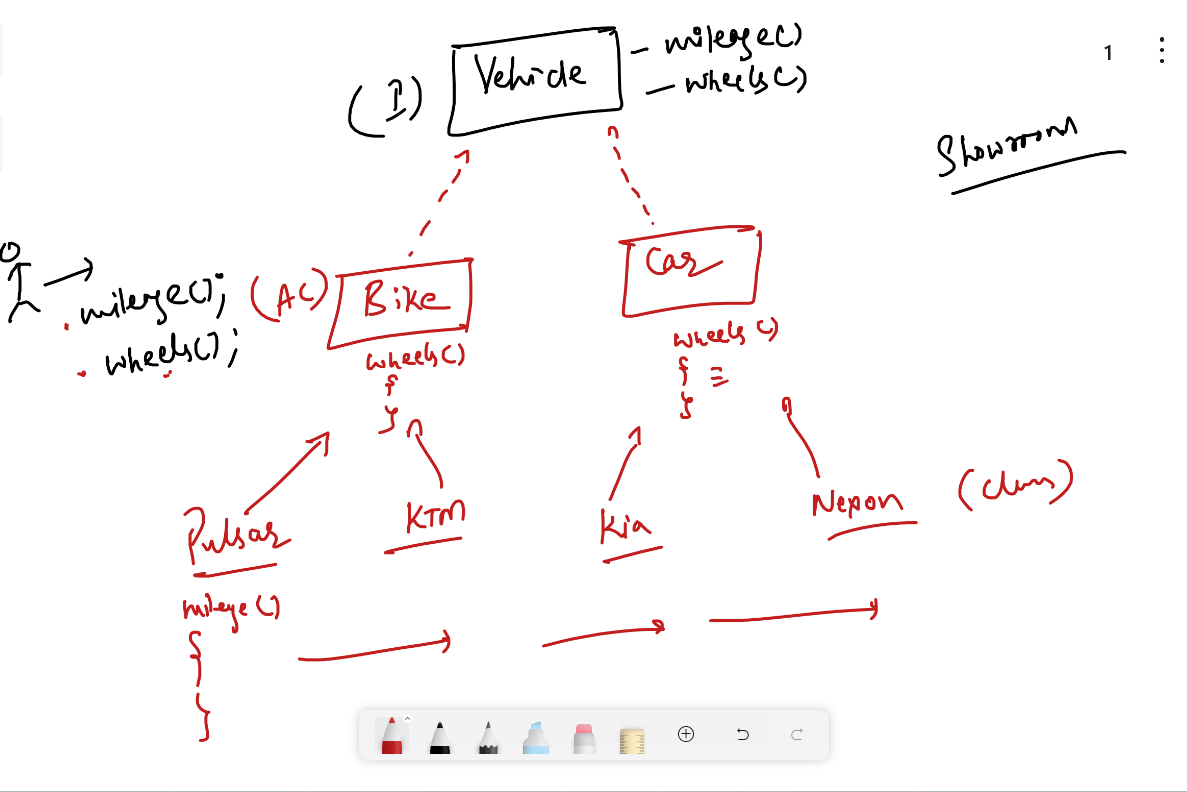
Important Note on interface

1. You can’t create object of interfaces
2. You can’t create constructors in interface
3. Methods are by default abstract
4. Variables are by default static & final i.e., constants

abstract class: It can have abstract methods & methods with body both, it can also have constructors, but you can’t create object of abstract class

abstract class Employee {   
 id, name, salary;  
 Employee(id, name, salary) { …. }  
 void display() { print id, name & salary }  
 abstract void incrementSalary();   
}

class Manager extends Employee {   
 // implement abstract methods – raiseSalary() – 25%  
}  
class Developer extends Employee {   
 // implement abstract methods – raiseSalary() – 20%  
}



TestAbstraction.java

