Java Fullstack

Course Agenda

* Core Java
* JDBC
* Javascript (ES new features)
* React.js
* Angular Framework
* Spring
* Spring Boot
* Spring Microservices
* DevOps and AWS

Core Java

Java is a platform independent programming language & it is object oriented as well.

Platform Independent: You can run java programs on any platforms without altering

Object Oriented: It allows you to create applications with real world entities which are called as objects, object will have 2 things mainly

1. properties (data/varaibles/fields)
2. behaviour (actions/functions/methods).

Software required

* JDK (Java Development Kit) - 1.8
* JRE (Java Runtime Environment) comes along with JDK
* Eclipse IDE (tools to develop java applications)

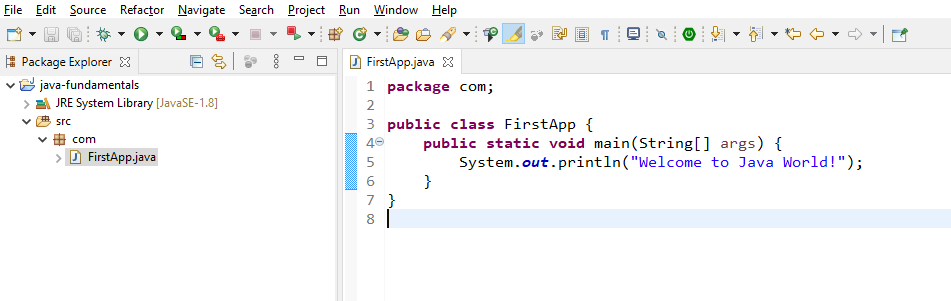
JRE: It is a runtime environment to run java applications, it will have JVM to execute byte codes

Building block of Object Oriented language

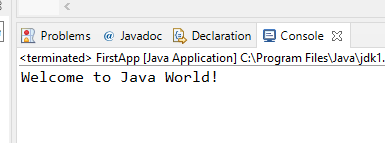
* Class: It is a template of an object, doesn’t take memory in the heap
* Object: It is a real world entity created from the class, takes memory in the heap

How to write our first java program

1. Use Eclipse & Create a java project, if in case you are in JavaEE perspective change it
2. Create a class & name it and also mention the package name
3. Create an entry point method i.e., main method
4. Run the main method.



Output:



Java Features

* Simple to understand, because most of the low level functionalities are abstract
* Object Oriented
* Platform Independent
* Architecture Neutral(x64, X86)
* Distributed applications
* Robust
* Secured
* Large Community
* Open Source

Fundamentals of Java

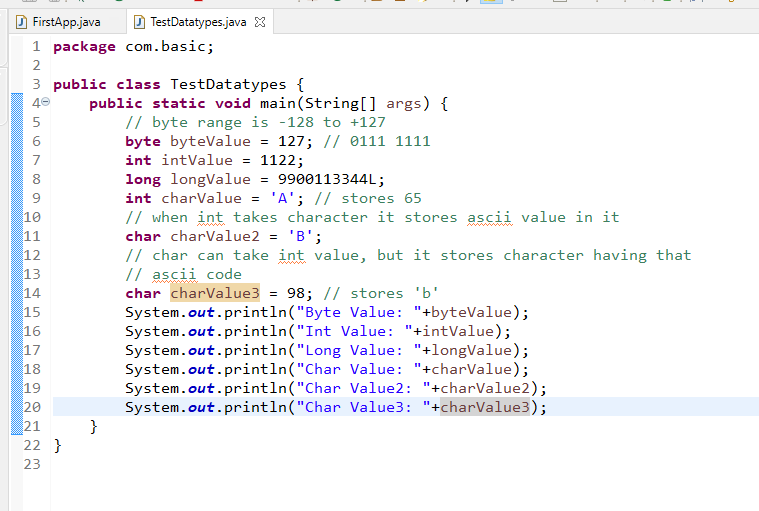
* Datatatypes
* Keywords
* Operators
* Conditional Statements
* Arrays
* Loops
* Variables
* Methods
* Classes
* Objects

Datatypes in Java

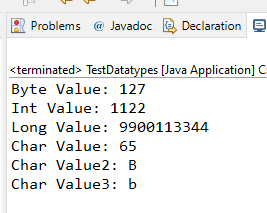
These are set of keywords used to create variables to store some value like numbers, characters, boolean, object and etc.

There are 2 types of datatypes in java

1. Primitive type: These are simple types whose size is defined
   1. byte = 1 byte
   2. short = 2 bytes
   3. int = 4 bytes
   4. long = 8 bytes
   5. float = 4 bytes
   6. double = 8 bytes
   7. char = 2 bytes
   8. boolean = 1 byte
2. Derived type: These are complex types that can store multiple values
   1. array
   2. class
   3. interface
   4. enum



Output:



Keywords in Java

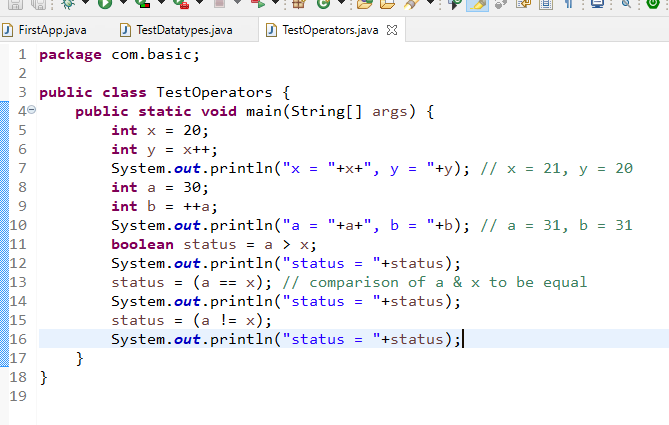
These are reserved words in Java, which must not be used to name the variables, methods or classes, they solve specific purpose

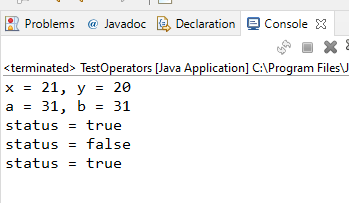
int, char, float, double, boolean, long, final, public, private, class, protected, static, void, return, if, for, do, while, switch, break, continue, abstract, interface, enum, extends, super, this and so on.

Operators in Java

+, -, \*, /, %, ++, --, =, ==, <, >, <=, >=, !=, &&, ||

TestOperators.java

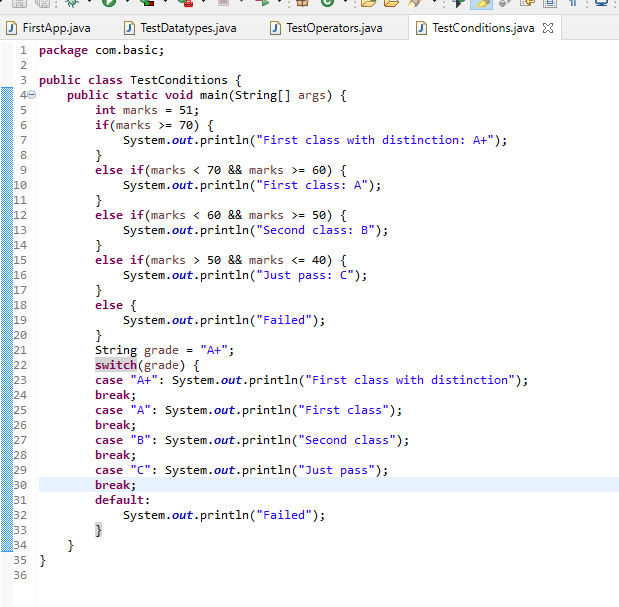




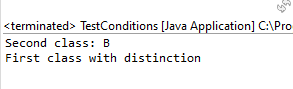
Conditional Statements

These are used when you want to execute the statements based on some conditions, we have following conditional statements

* if
* if else
* if else if .. else
* switch



Output:



Scanner class:

It is a predefined class present in java.util package, which is used to take input from the keyboard, you need to import this class using ‘import java.util.Scanner’

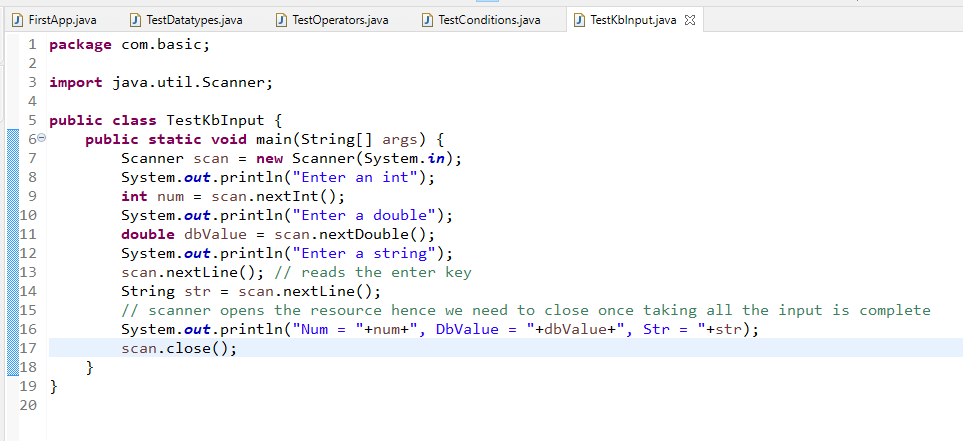
It defines lot of methods to accept different types of value like

nextInt(), nextLong(), next(), nextFloat(), nextDouble() and so on

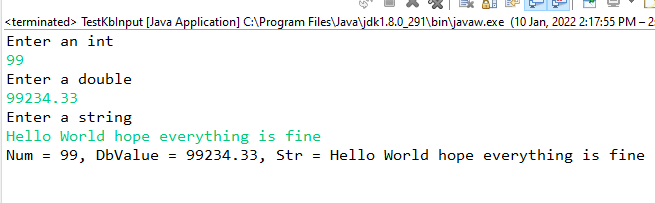
All the above methods you can call by creating scanner object

Scanner scan = new Scanner(System.in);

The above code creates object of scanner, System.in is a reference to the input stream i.e., keyboard



Output:



How to work with arrays

In Java you can create arrays using [], you can create arrays of simple types to complex types

Different ways of initializing array & their values

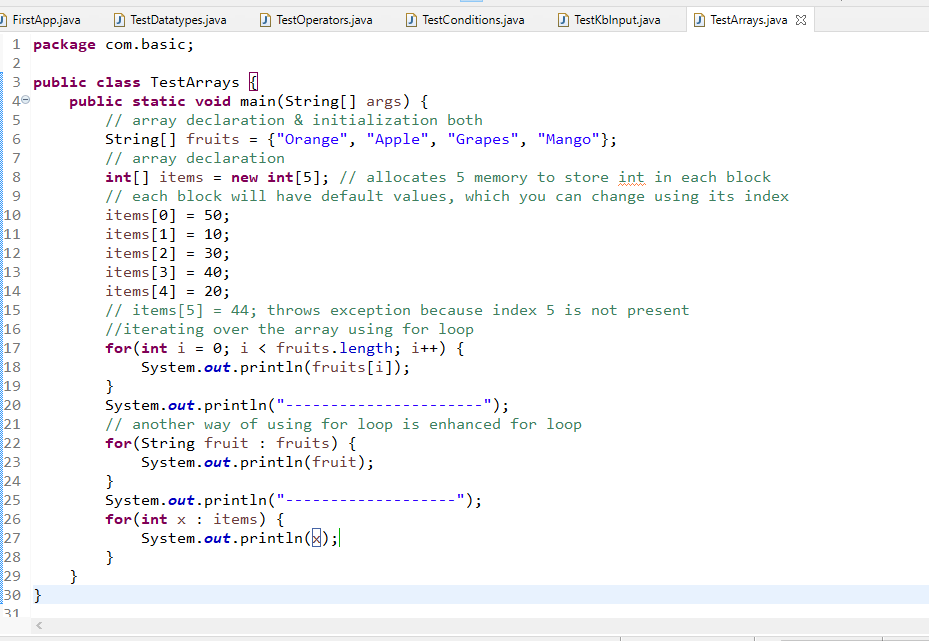
int[] items = {10, 20, 15, 35, 30}; // it is an integer array which is initialized

int[] items = new int[5]; // it is an integer array but not initialized the blocks

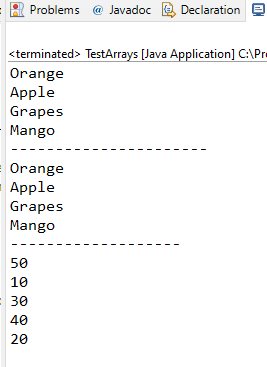
items[0] = 10; items[1] = 30, … items[4] = 50;

String[] fruits = {“Apple”, “Mango”, “Orange”}; // string array which is initialized

You can use loops to iterate over each items of the array i.e., for loop



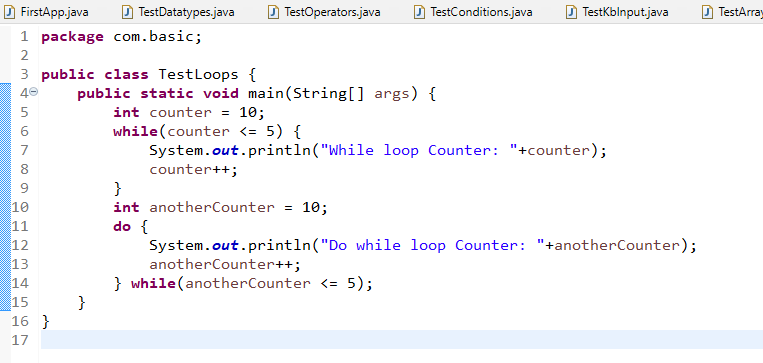
Output:



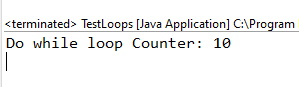
Loops in Java

We have 3 types of loops

1. for: When you want to iterate fixed number of times or you know how many times you want iterate
2. while: When you want to iterate until condition is true
3. do-while: When you want to iterate until condition is true, but in do while the statement is atleast executed once



Output:

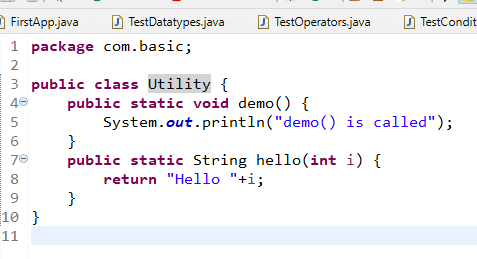


Methods in Java

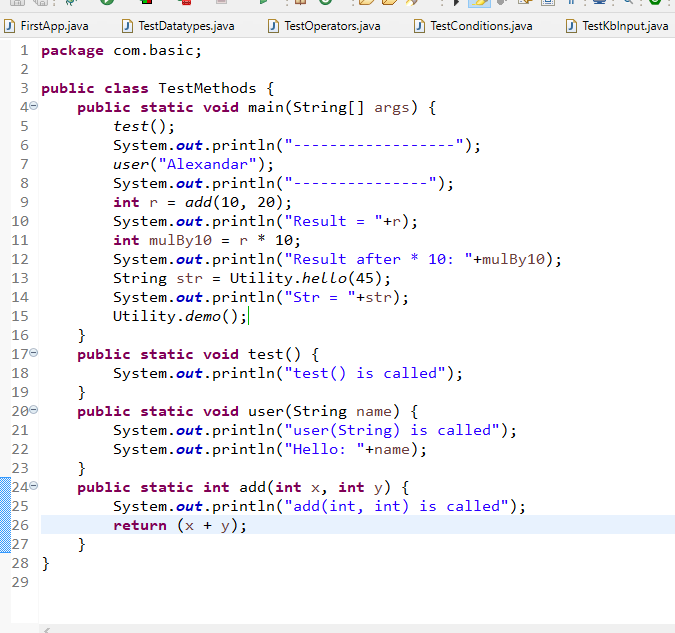
You can create methods in class in 2 ways

* Using static keyword: You can call this method without creating object
* Without static keyword: You can call this method by creating object

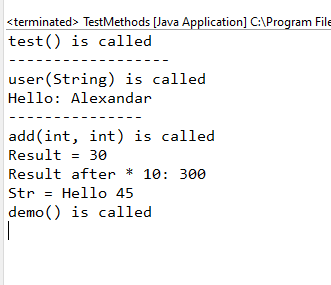
Utility.java



TestMethods.java



Output:



Activities

Create a Exercise folder & day1 sub-folder & keep all the day1 activities here, same way on 2nd day you will create day2 sub-folder and so on.

1. Try all the above examples
2. Create a method that accepts name as a parameter & returns a welcome message, call this method from the main method and display the welcome message

Ex: If you enter Raj the method should return Hello Raj, welcome to java

1. Create a method that accepts 3 numbers as parameters & returns the largest number among them

Ex: If you pass findLargest(3, 9, 1), then it must return 9

1. Create a method to accept 3 numbers & returns the sum of biggest & smallest number.

Ex: If you pass sum(9, 1, 3) then it must return (9 +1), if you pass sum(1, 2, 5) then it must return (5 + 1)

1. Create a method that accepts 3 numbers & prints their digits in words ex: If the input is 754 then the output should be “Seven Five Four”

Day 2 Agenda

* Method Overloading
* Classes & Objects
* OOPs concepts
* Static & Final Keywords
* String, Integer, Float, Double, Object

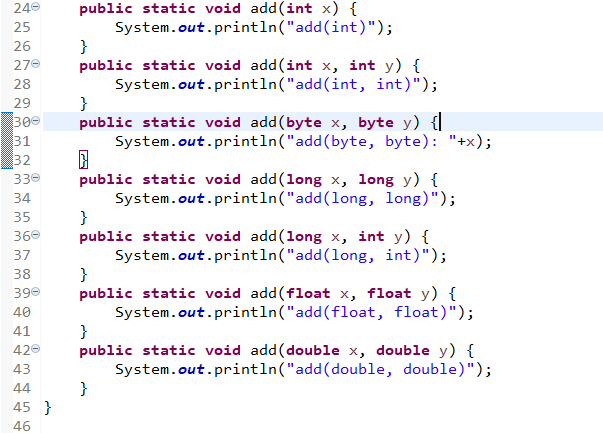
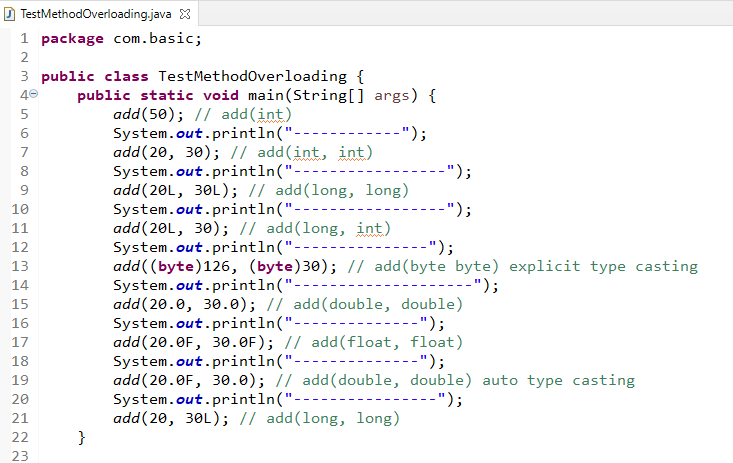
Method Overloading:

Multiple methods with the same name but different parameters or different signature, Compiler can identify which method needs to be called, hence it is also called as compile time polymorphism

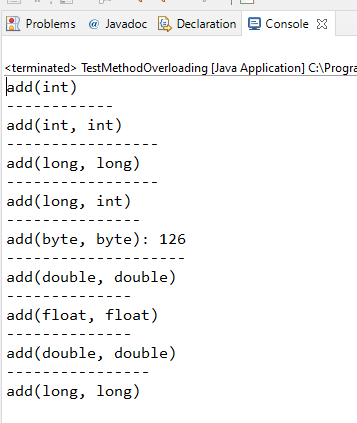
Polymorphism: Poly means many and morphism means forms, overloading is one of the type of polymorphism, there are 2 types of polymorphism

* Method overloading: Compile time polymorphism, methods are identified at compile time itself.
* Method overriding: Runtime polymorphism: Methods are identified at runtime.

TestMethodOverloading.java



Output:

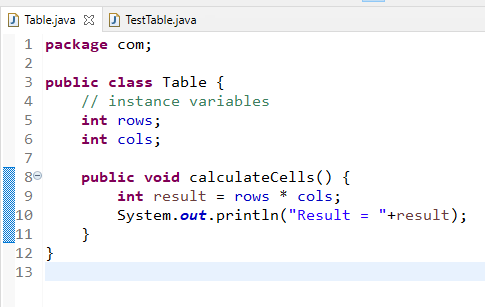


Classes & Objects

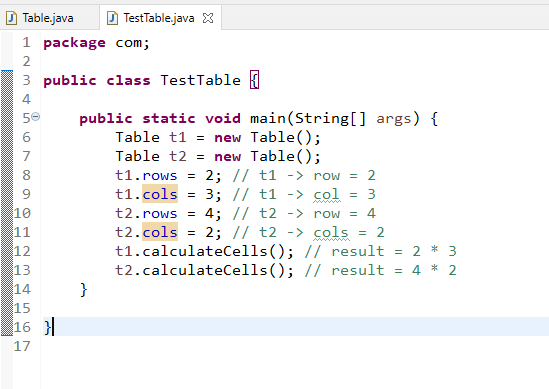
Class is a blueprint of an object, which describes the object

Object is an instance of the class.

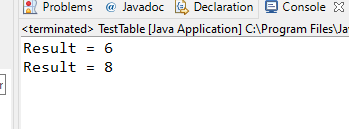
Table.java



TestTable.java



Output:



The above code is initializing the variables of each new object of Table, in a Table class we don’t have any constructor, but compiler creates a default constructor.

Constructor: It is a like a method but name will be same as class name & it should n’t have return types, In Java if you create a class *compiler automatically creates a default constructor if in case user has not created any constructor.*

For ex:

class Test {   
  
}

The compiler creates a default constructor as below

class Test {  
 Test() { …. } // default constructor  
}

Another ex:

class Demo {  
 Demo(int x) { … }  
}

Now the compiler retains the constructor user has provided

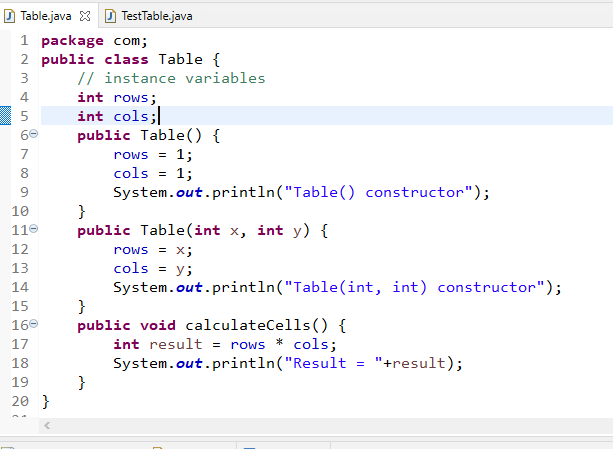
class Demo {  
 Demo(int x) { … }  
}

Another ex;

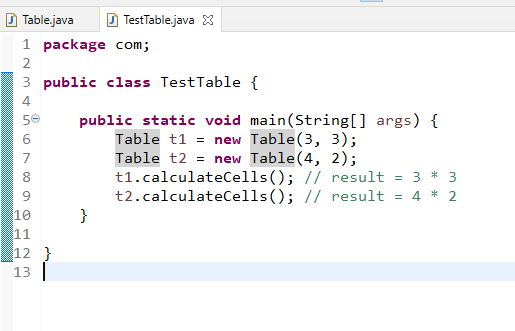
class Hello {  
 Hello() {..}  
 Hello(int x, int y) {…}  
}

Now the compiler retains both the constructors, it means you can also have multiple constructors which means constructor overloading.

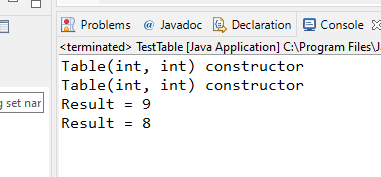
Table.java



TestTable.java



Output:

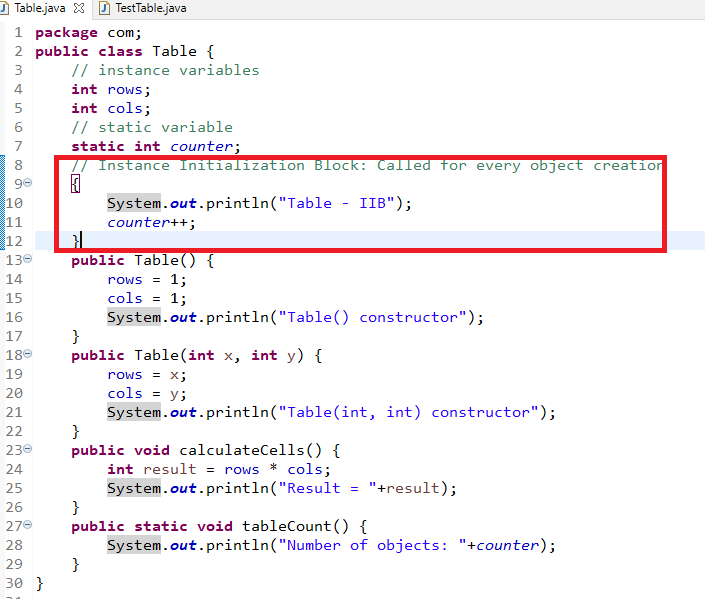


The above code uses parameterized constructor to initialize the instance variables, which also avoids lot of initialization statement at the caller side.

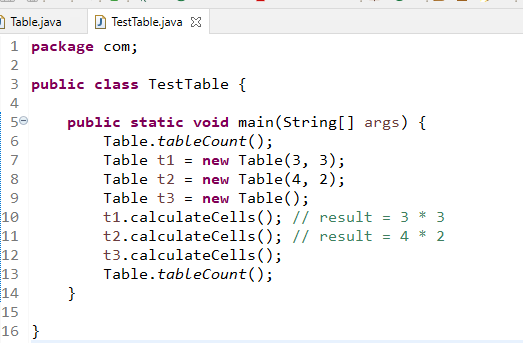
Instance Variables: These are the variables which are created for every object

Static Variables: These are the variables which are shared for all the objects, it will be a single copy created for all the objects of the class, you need to access static variables through class name though you can access it using reference variable, its recommended to access static variables or methods through class name, reason is compiler converts the reference type to class name

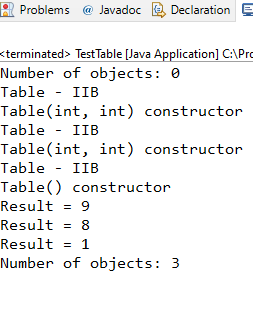
Table.java



TestTable.java



Output:



Features of OOPs

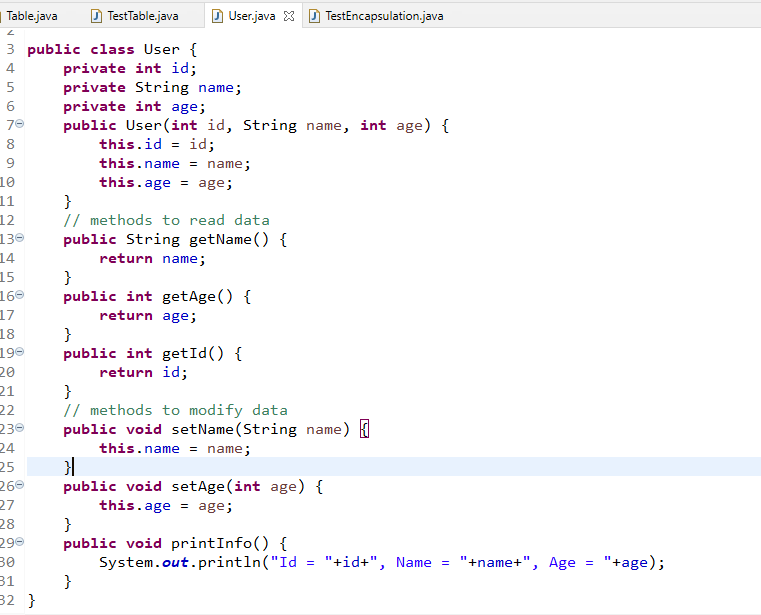
* Encapsulation
* Inheritance
* Polymorphism
* Abstraction

Encapsulation:

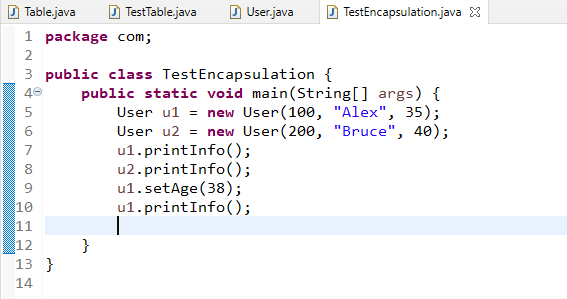
Binding the data & methods in a single entity, you will make data private so that it is accessible only within the class & you will have methods for each variables to read/write which are called getters & setters.

Through encapsulation you will have more control over the data, you can avoid invalid values and also avoid accessing the data if not required.

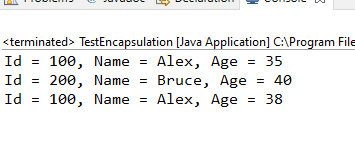
User.java



TestEncapsulation.java



Output:



In Eclipse you have an easier way to generate setters & getters as well as constructor also.

Right Click -> Source -> Generate Setters & Getters

Inheritance

When you want to acquire properties & behaviours of an object from another object you can use inheritance, in java you use extends keyword to achieve inheritance.

class A { }

class B extends A { }

Here A is a super class or parent class whereas B is subclass or child class

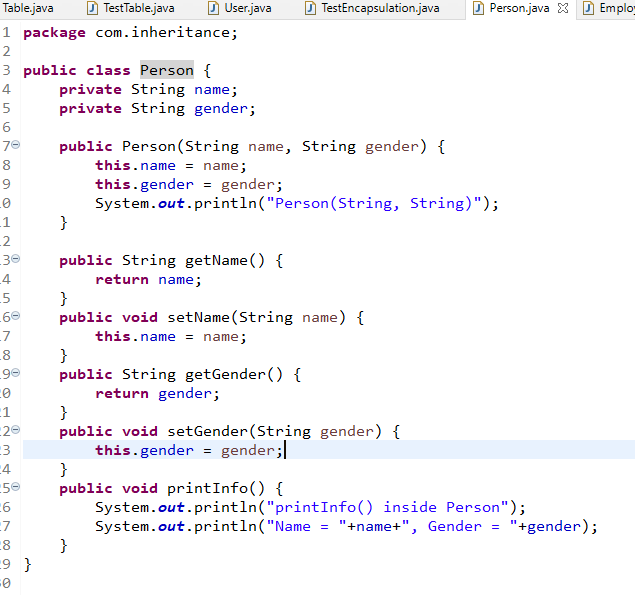
Note: Private members & Constructors wouldn’t be inherited

Note: You can use super keyword to access super class members & constructors

super.member // it accesses super class members from sub class

In Inheritance every constructor of the subclass calls the parent class default constructor from its first line implicitly, you can explicitly call the parameterized constructor of the parent class using super(args) on the first line of the subclass constructor

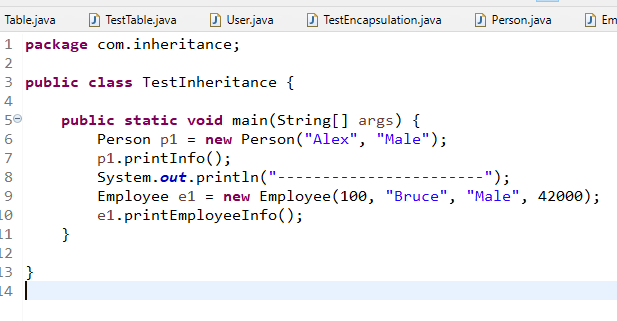
Person.java



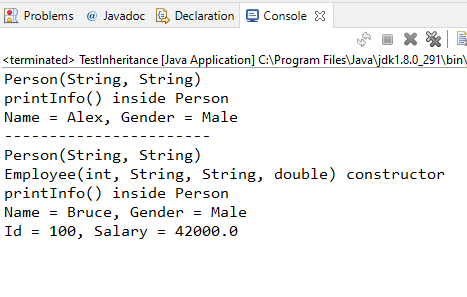
Employee.java



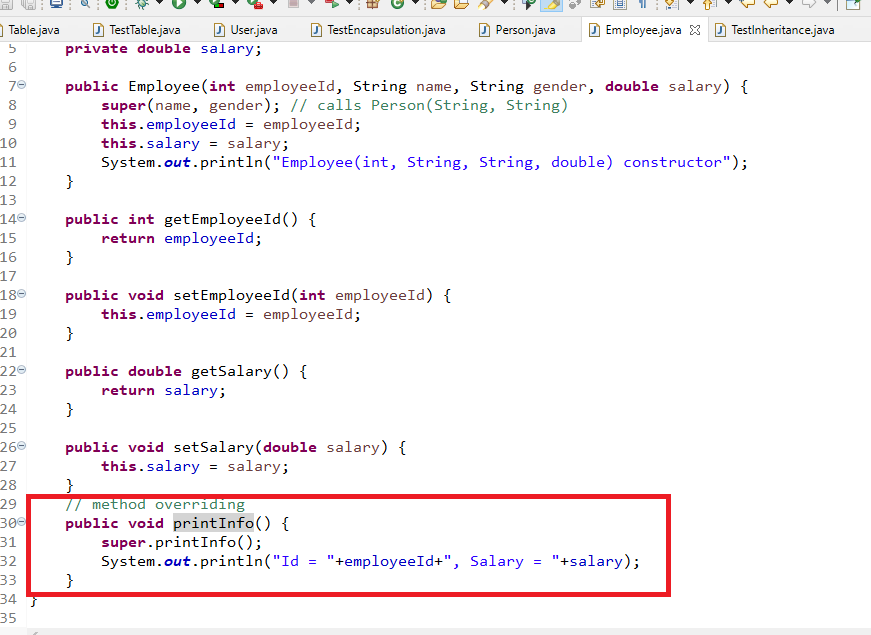
TestInheritance.java



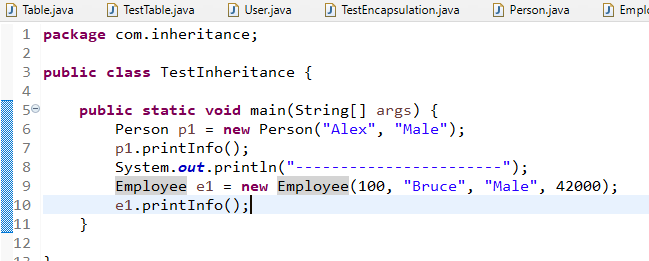
Output:



Instead of creating different methods in subclass to print the data you can override printInfo



Now if you call printInfo on employee object it calls the printInfo present in the Employee object.

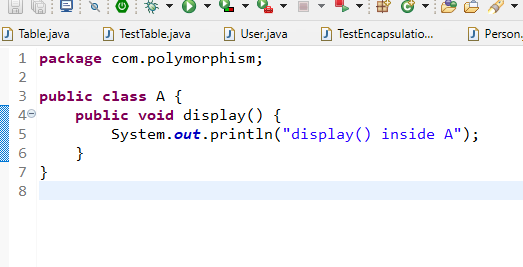


Polymorphism

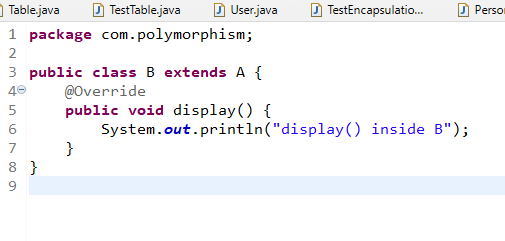
Ability of a method to give multiple results

ex: Single Power button acts like On and Off both, similarly single method can give different results based on the object you are using.

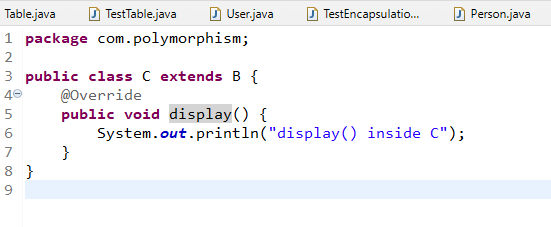
A.java



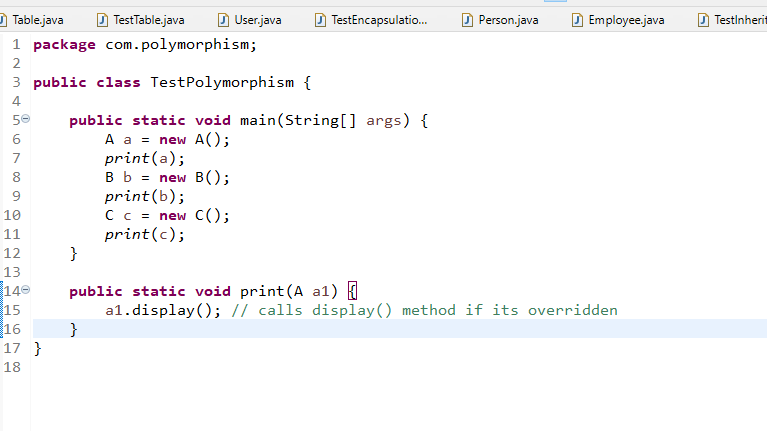
B.java



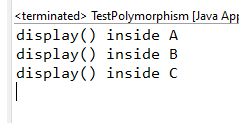
C.java



TestPolymorphism.java



Output:



Activity:

1. For the above inheritance & polymorphism example create display method in Person & override in Employee to print employee information
2. For the same example create a Student & Customer class that should extend Person,
   1. Student class must have rollNo & grade properties & Override display method to print rollNo, name, gender & grade
   2. Customer class must have customerId, accountNumber, balance & Override display method to print customerId, name, gender, accountNumber & balance
   3. From main method create Object of Student, Customer, Employee & Person and pass these objects to a single static print method that all the subclass of person like Customer, Employee, Student and Person also

ie., print(employeeObject), print(studentObject) and so on, it must call display method to print employee, student, customer informations.

1. Define a class Student with the following attributes

* studentId of type integer
* studentName of type String
* city of type String
* marks1 of type integer
* marks2 of type integer
* marks3 of type integer
* feePerMonth of type float
* isEligibleForScholarship of type boolean

Implement the following methods in addition to the setter and getter methods for the various attributes

* getAnualFee() which returns the product of feePerMonth and 12
* getTotalmarks() which returns the sum of marks1, marks2 and marks3
* getAverage() which returns the average of marks1, marks2 and marks3
* getResult() which returns “pass” if the person has scored more than 60 in each subject, or returns “fail” otherwise

Create another class TestMain with the main() method which performs the following actions

* Creates three Student objects
* Populates the objects using the setter methods
* Displays the name of the Student who has the highest total marks
* Prints the name and fee of the Student who pays the least monthly fee
* Prints the name, total marks , average marks , result, and “Scholarship available” or “Scholarship not available” based on the student’s eligibility for every student.

Abstraction

It is a process of hiding the complexity form the user and showing only the necessary details which the user can use without knowing the complexity.

In Java you can achieve abstraction in 2 ways

1. Interface: Fully abstraction
2. Abstract class: Partial abstraction

Interface: It is a kind of class where you will have only abstract methods and constants

abstract methods: These are methods without logics, it will have only method declaration

Interface methods needs to be implemented by classes using implements keyword

Ex:

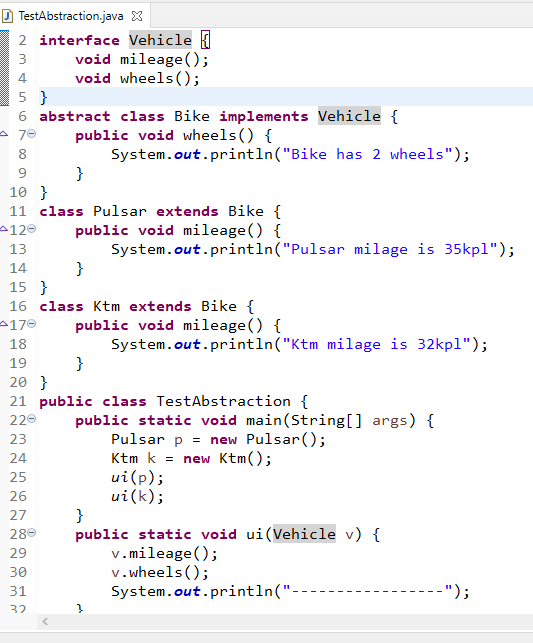
interface A {   
 int x = 10; // by default variables are final & static  
 void test1(); // by default methods are abstract  
 void test2(); // by default methods are abstract  
}

Note: All the members in the interface take public keyword

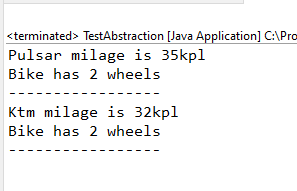
class X implements A {   
 // Now X must implement test1 & test2 mandatorily  
}

abstract class Y implements A {   
 test1() { …… }  
}

TestAbstraction.java



Output:



Activity:

Create an abstract class Car and implements wheels method of Vehicle, then create two classes that extends Car

* Swift >> implement milage
* Ferrari >> implement milage

Pass Switch & Ferrari object to the ui method that takes these objects and calls mileage() & wheels()