Java

Java is Platform independent & Object Oriented language

Platform Independent: You can run your code on any platform without recompiling

Object Oriented: You can create applications that will have real world entities which are objects.

Object will have properties & behaviours

Ex:

Customer

properties: id, name, dob, gender, ....  
 behaviours: withdraw(), deposit(), ..

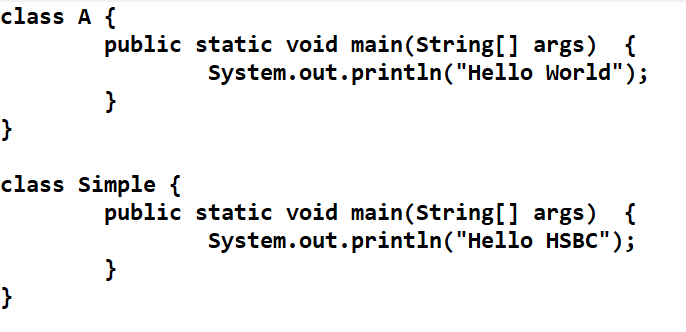
Basic Building block of OOPs

1. class: Blueprint of an object
2. object: Instance of a class

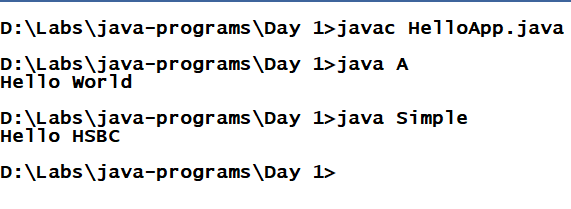
Some of the java commands:

1. javac
2. java
3. javap
4. jar

HelloApp.java



Output:



Fundamentals of Java

1. Datatypes
2. Operators
3. Conditions
4. Arrays
5. Loops
6. Keywords
7. Class & Objects

Datatypes:

Describes a variable can store what kind of value, there are two types

1. primitive type
2. derived type (user-defined type)

Primitive Datatypes:

|  |  |  |  |
| --- | --- | --- | --- |
| Integers | Floats | Boolean | Character |
| byte (1)  short (2)  int (4)  long (8) | float (4)  double (8) | boolean (1) | char (2) |

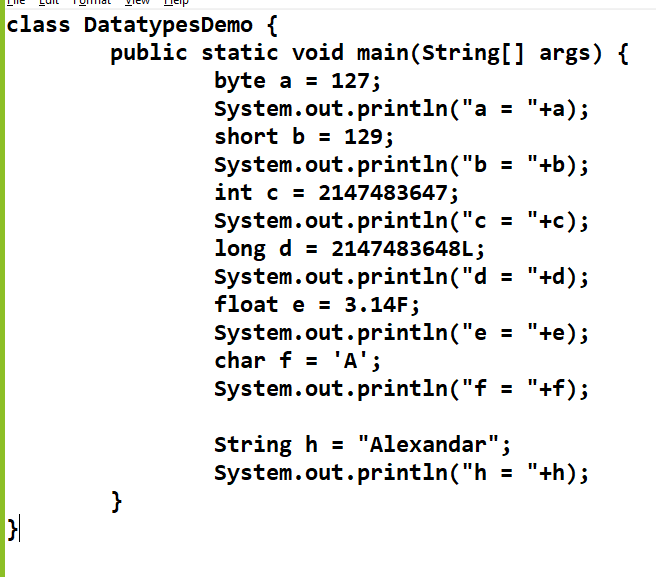
Derived Datatypes:

class, interface, array

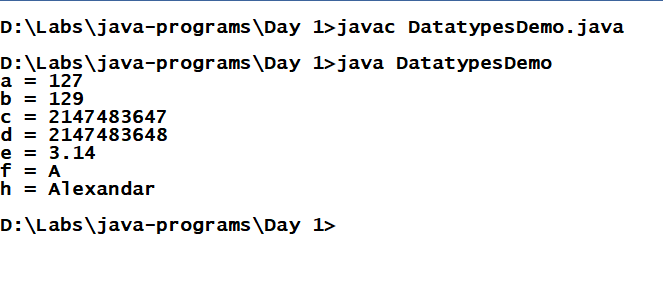
Operators:

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

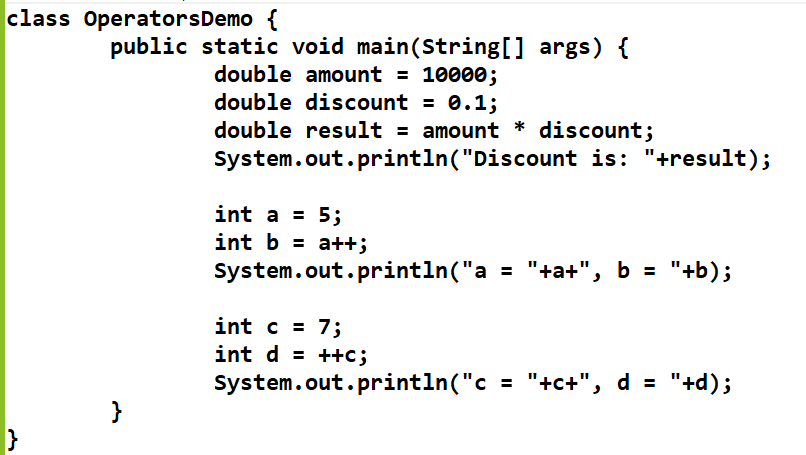
TestDatatypes.java



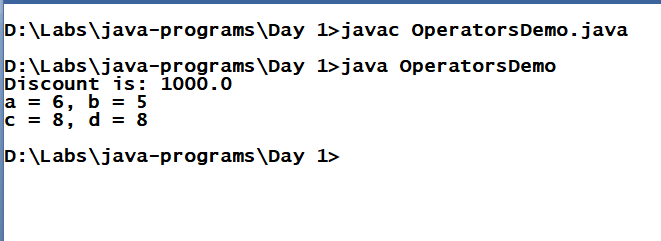
Output:



OperatorsDemo.java

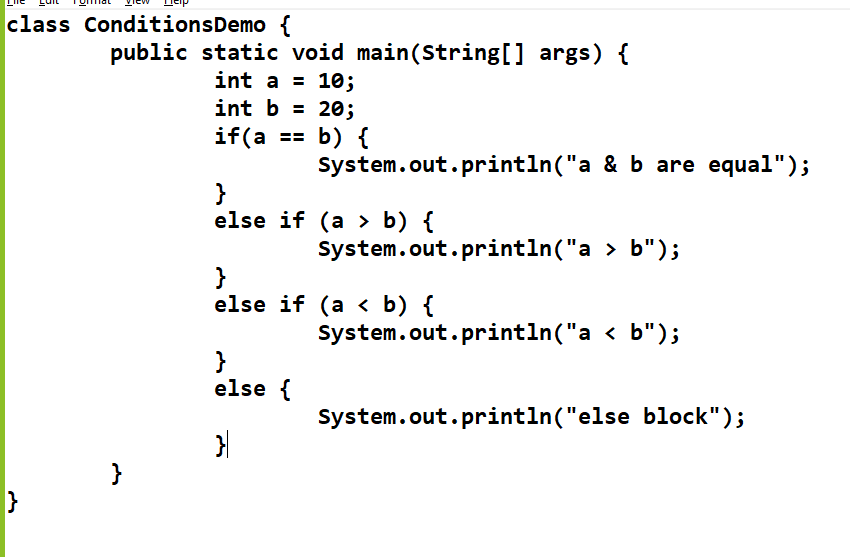


Output

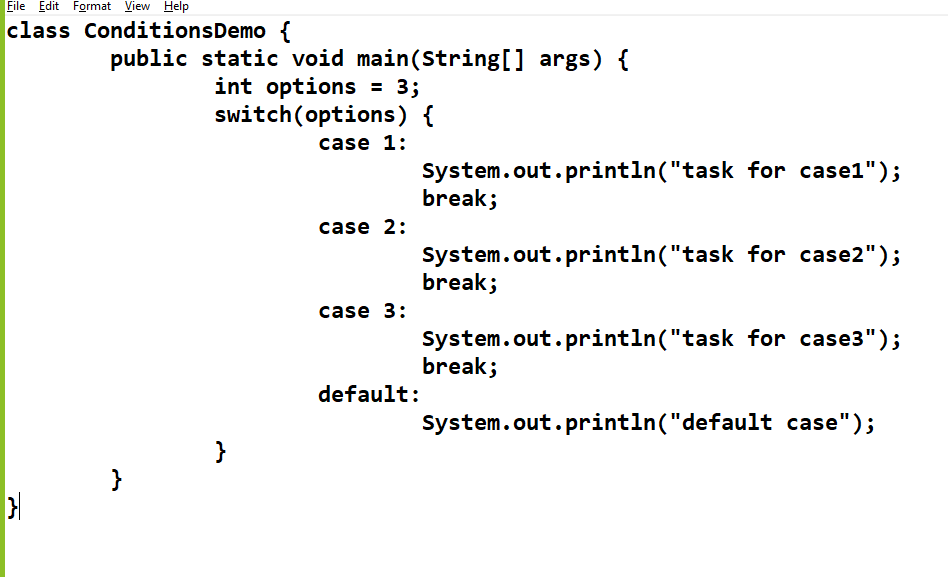


Conditional Statements

1. If
2. If Else
3. If Else If.... Else
4. Switch



Switch

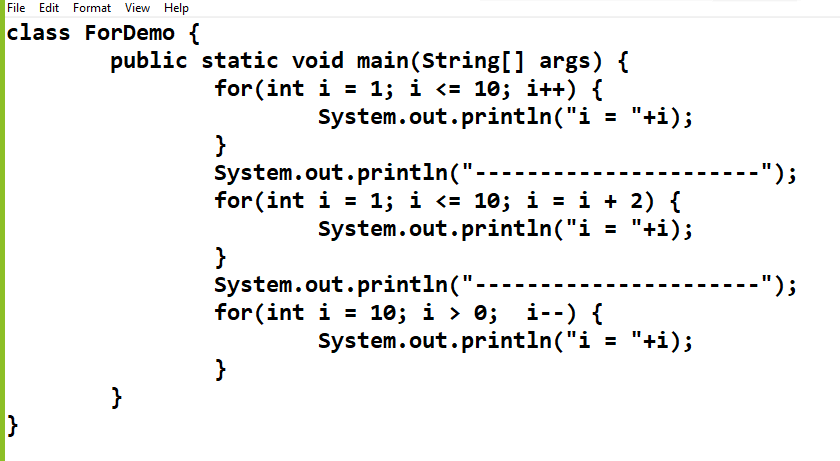


Note: Switch can be used only for numbers, strings, char & enum

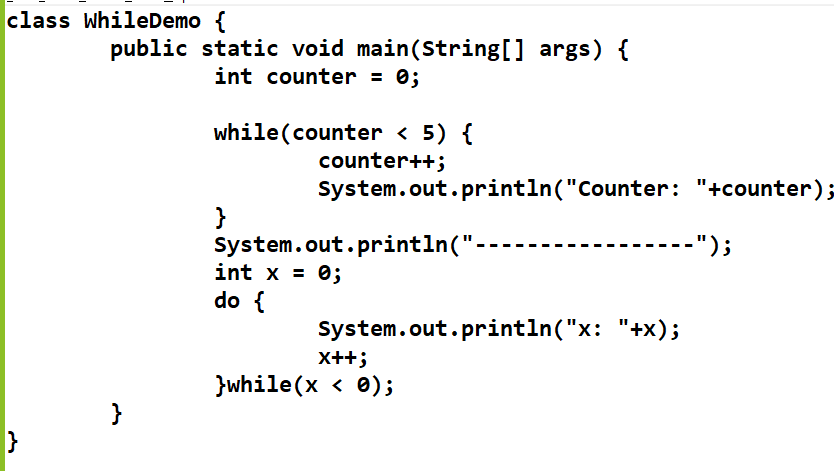
Loops:

1. For
2. While
3. Do-While

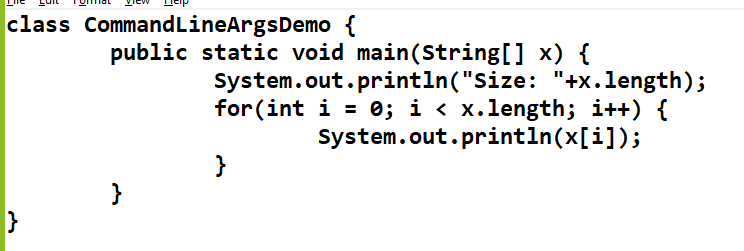
ForDemo



While & Do While Demo



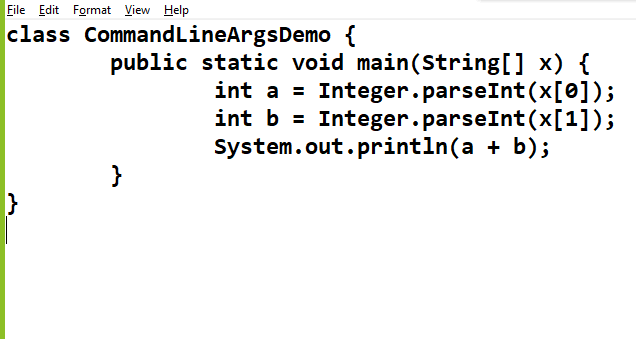
main(String[] args);



Command line argument is string type, you can convert to int or double or char or boolean using some inbuilt classes

Integer.parseInt(“10”); returns int

Double.parseDouble(“10.25”); return double



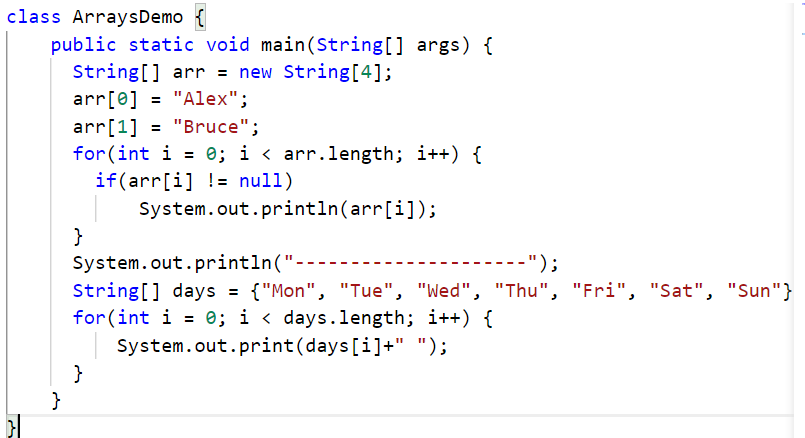
Arrays:

It stores multiple items of same types

Type[] variable = new Type[size];

Type[] variable = {value, value, .....};

ArraysDemo.java



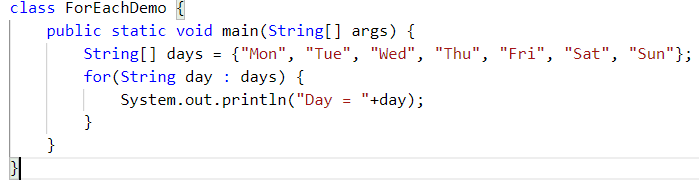
for each loop:

This is introduced in Java 5, to simplify writing the traditional for loop, it doesn’t need any variable initialization, comparison, increment statement, index access

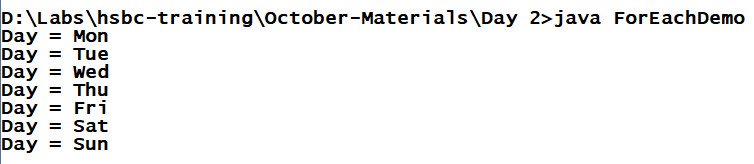
Syntax:

for(Type t : collection) { ... }

collection could be an array or any collection



Output:



Enhanced for loop:

* It iterates only in forward direction
* It iterates all the items, if you want to iterate only few items you must write extra logics

classes & objects

Class: It is a template of an object

Object: It is a real world entity or it is an instance of a class

Classname t = new Classname();

Inside class you can have

1. variables
2. methods
3. constructors

Variables are fields to store state of an object

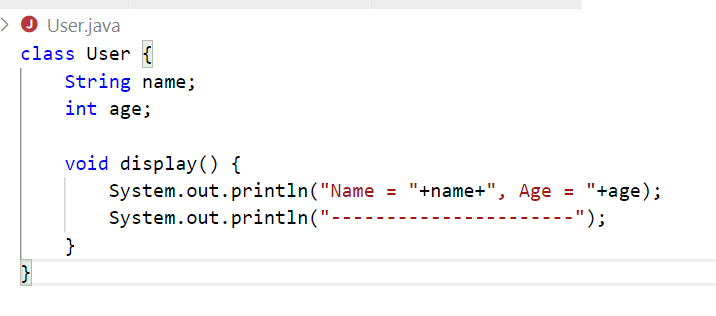
Methods will have logics

What are constructors?

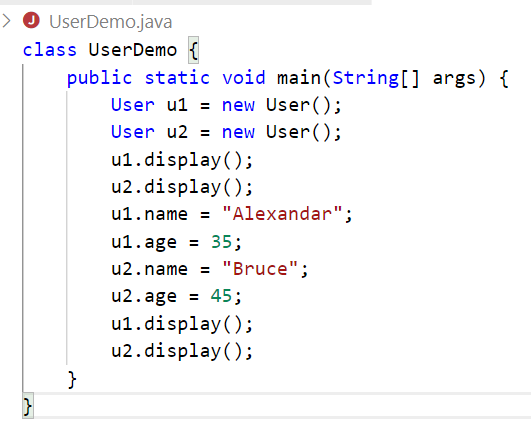
* Constructors are called when object is created
* Their name will be same as class name
* They will not have return types
* By default compile creates a constructor if class doesn’t have any constructor, which is called as default constructor
* If you provide a constructor in a class then compiler retains what is provided, it doesn’t add any default constructor

javap command can be used to check the informations of the class file

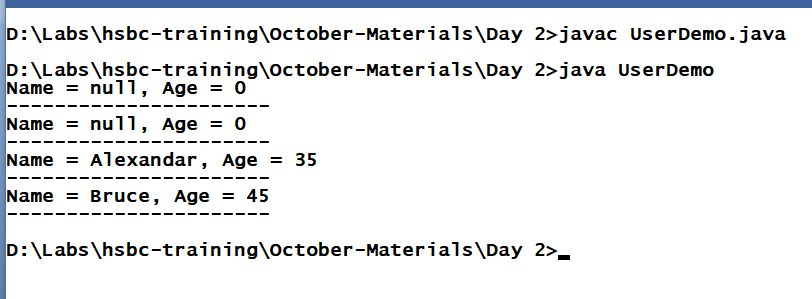
User.java



UserDemo.java

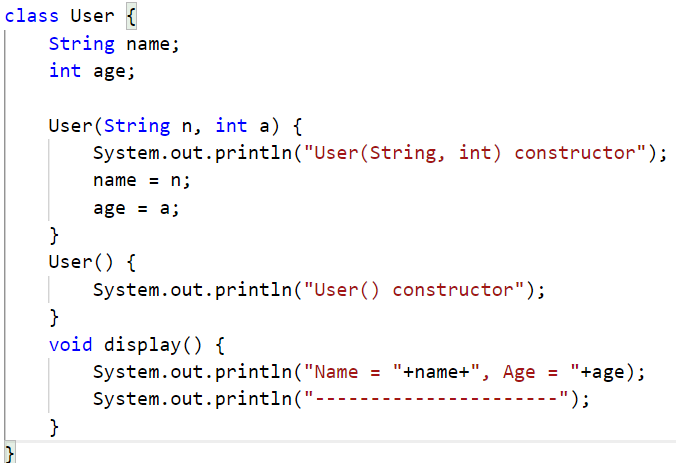


Output:

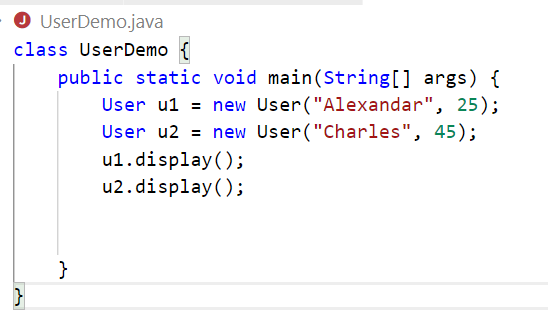


Constructor Overloading: You can have constructors with different parameters in the class, it initializes the variables at the time of object creation

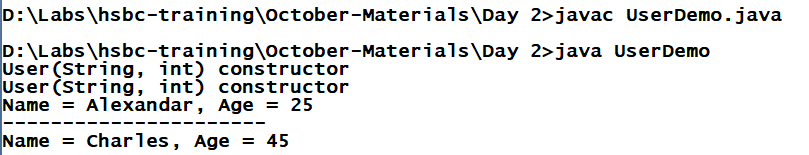
User.java



UserDemo.java

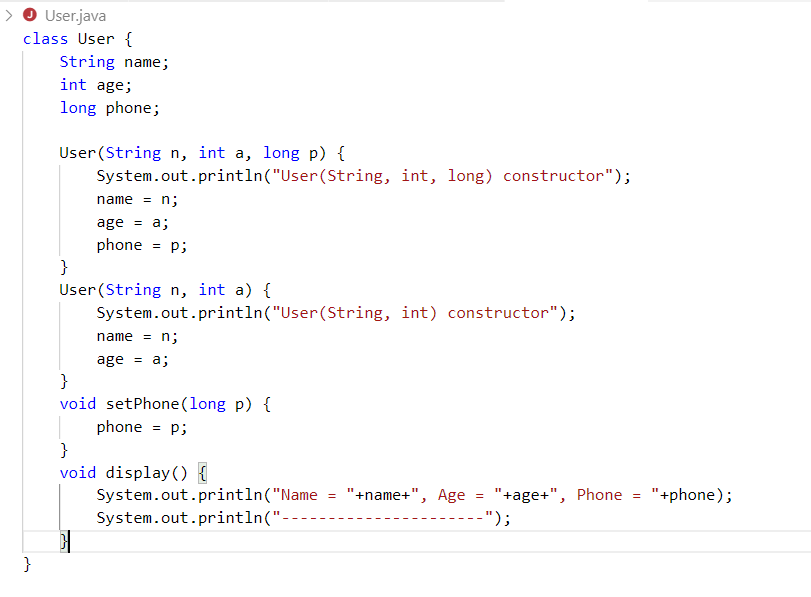


Output:

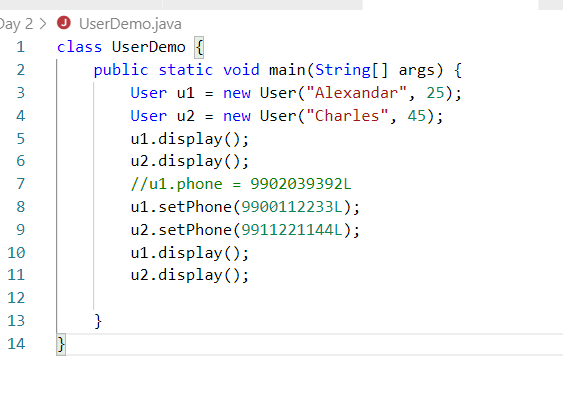


You can also initialize only few members

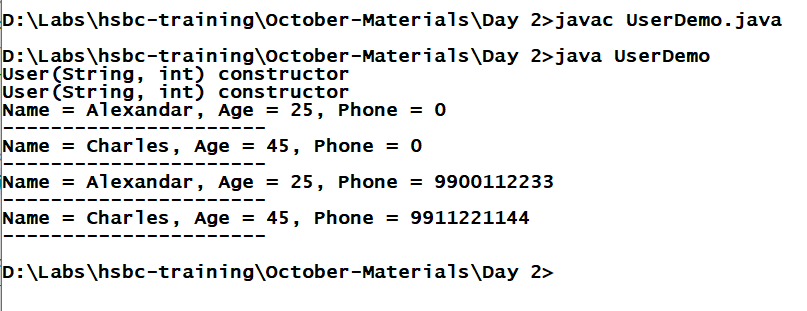
User.java



UserDemo.java



Output:



Assignment:

Create one savings account class and provide multiple constructors,

* Fields: accountNo, name, balance, ifsc, state, city, pin
* Methods: deposit(), withdraw(), display()
* Constructors:
  + Initialize accountNo, name, state, city & default value for ifsc=”HSBC001”, balance 0.0
  + Initialize accountNo, name, balance, state, city & default value for ifsc=HSBC001
* Create 5 objects for different constructors & call withdraw(), deposit() & display()

Static Members:

You can make variables & methods static

* static variables: common data for all the objects
* static methods: common logics for all the objects
* static members are loaded at the time of class loading
* static members can be accessed through class name or reference variable

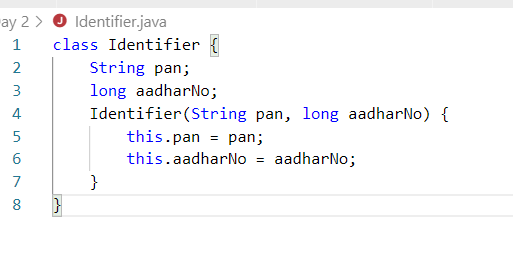
Note: Accessing through reference variable is not recommended, compiler converts reference variable to its class-name

this: It is a reference to the current object, you will use this to differentiate between instance variable & local variable when they share same name,

Another use of `this` keyword is call the constructor from another constructor

Pass by reference

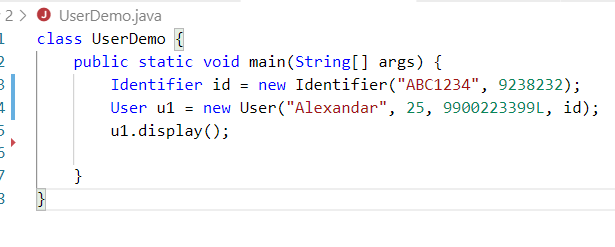
Identifier.java



User.java



UserDemo.java



Assignment:

1. Create an array of type Employee and store employee objects into the array, display only those employees who has salary > 50000 through for each loop.

Note: Atleast have 5 employee objects

1. Modify the Employee class to have Address(state, city, pin) variable with a Has-a relationship, display the employees with address

Note: You must have a constructor to take Address also as a parameter

OOPs concepts

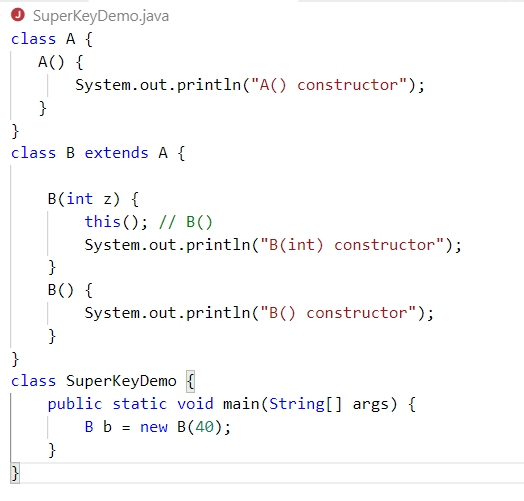
1. Inheritance
2. Polymorphism
3. Encapsulation
4. Abstraction

Inheritance:

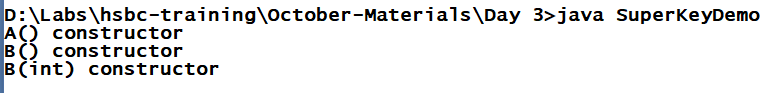
Process acquiring members from parent class to child class.

* extends keyword is used
* super class *default constructor* is automatically called from *subclass constructor*
* super() calls default constructor, super(args) calls argument constructor

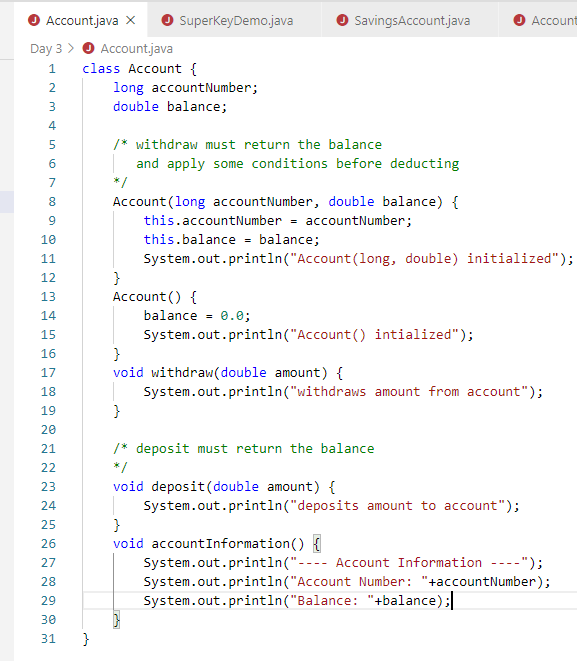
Inheritance forms a is-a relationship



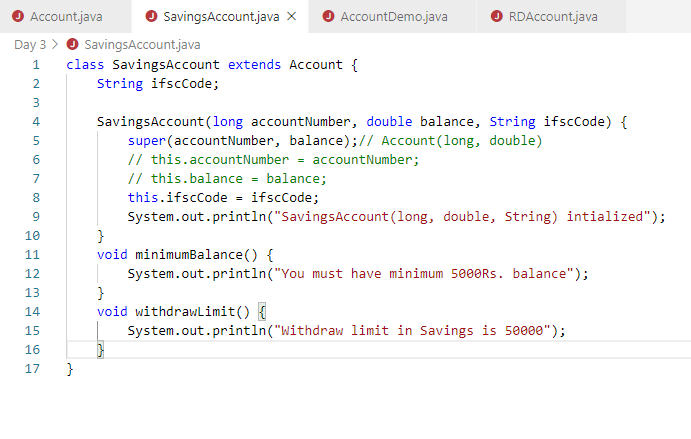
Output:



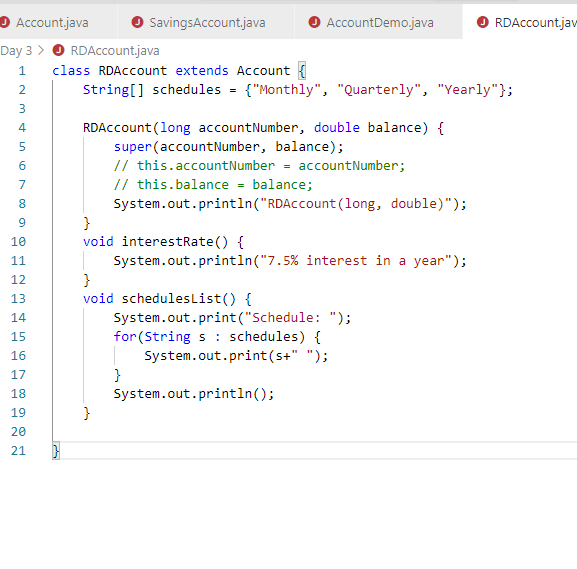
Account.java



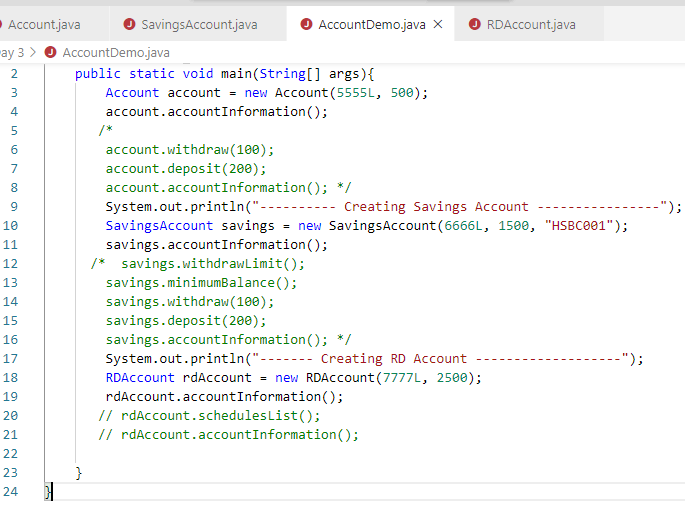
SavingsAccount.java



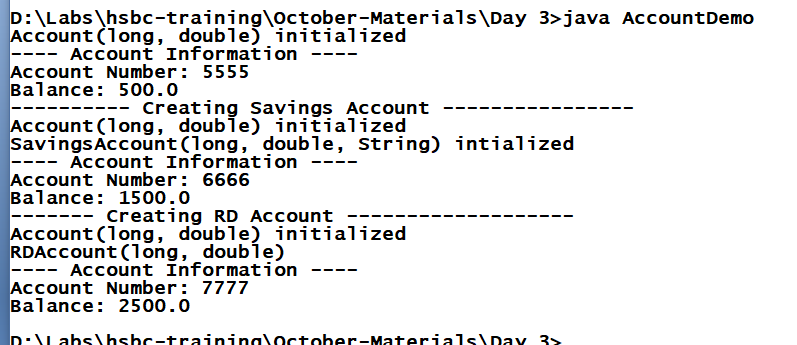
RDAccount.java



AccountDemo.java



Output:

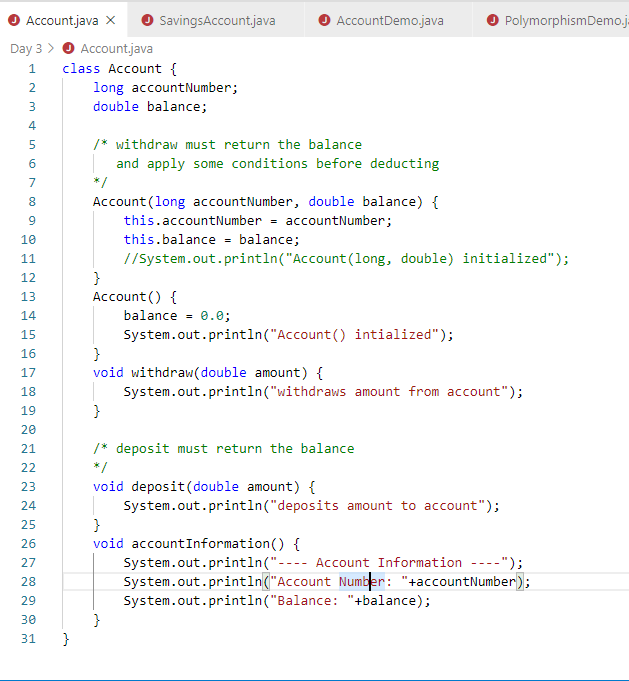


Polymorphsim:

Polymorphism means ability of a method to have many forms

1. compile time - method overloading
2. runtime - method overriding

Account.java



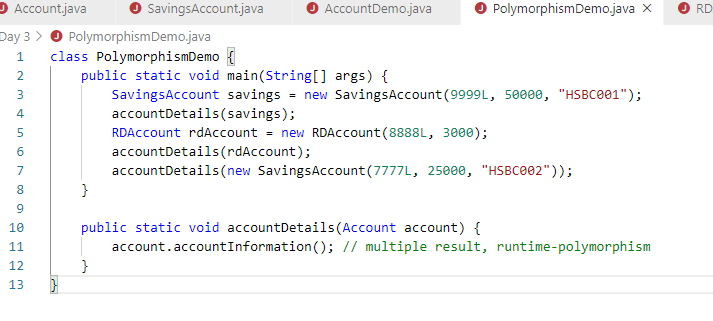
SavingsAccount.java



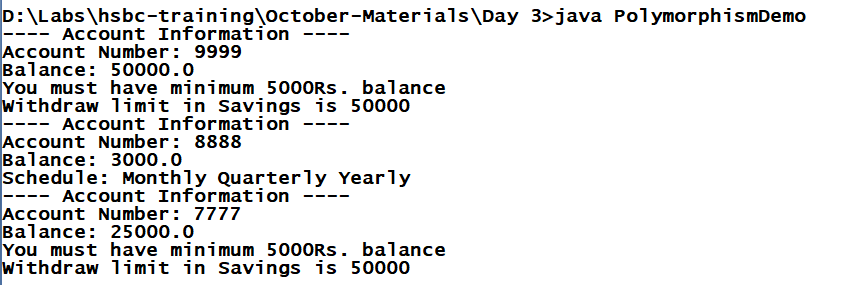
RDAccount.java

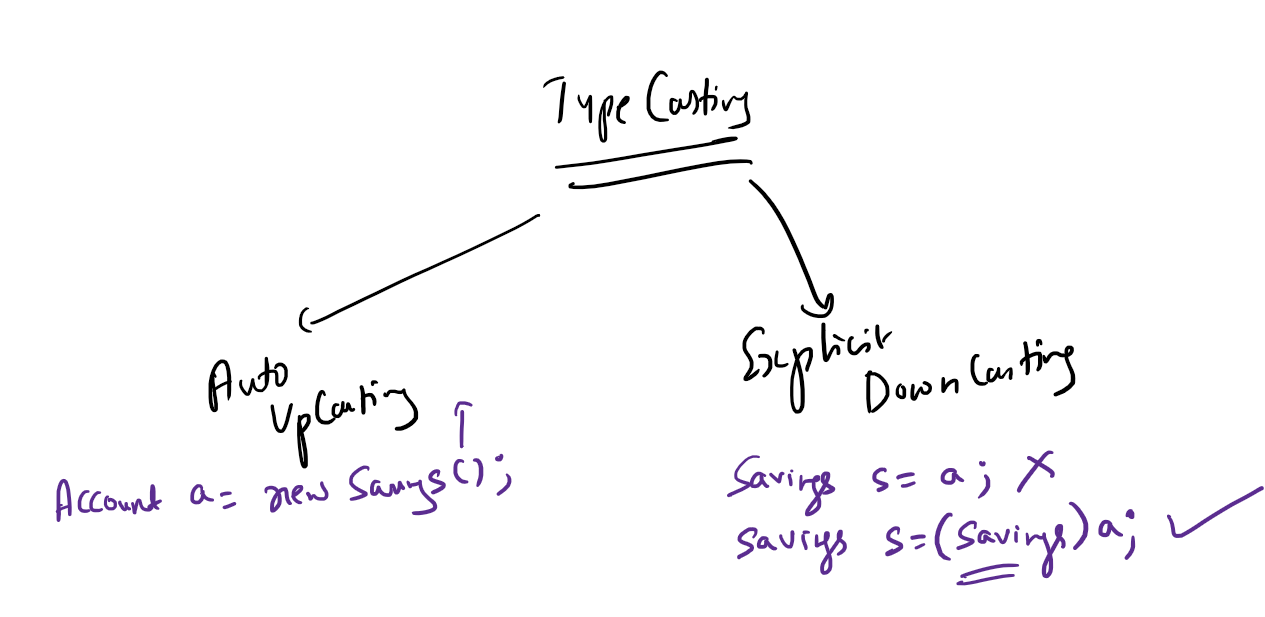


PolymorhismDemo.java



Output:

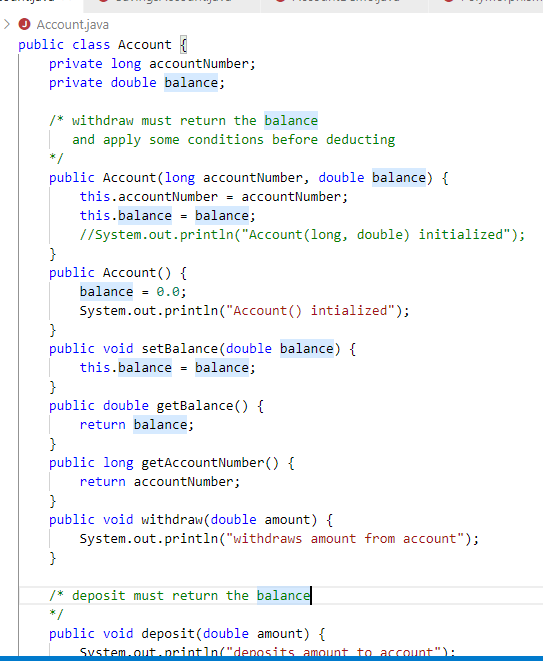




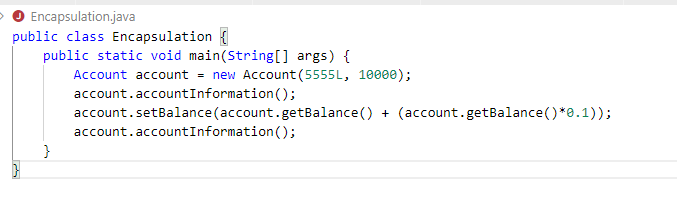
Encapsulation:

Hiding the data and accessing through public members, it gives control over the data.

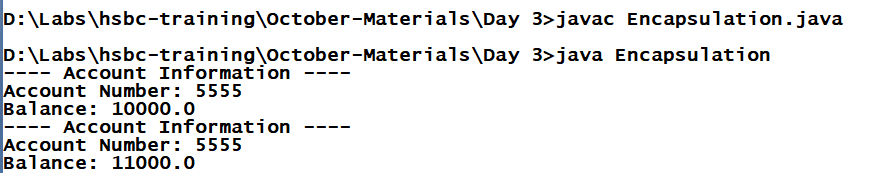
Account.java



Encapsulation.java

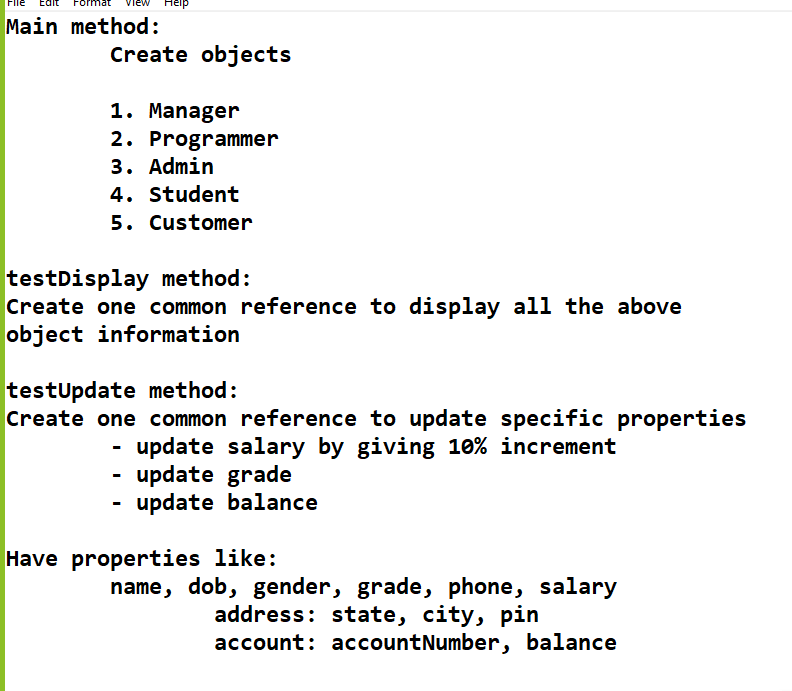


Output:



Abstract class & abstract methods:

When you don’t want a class to be instantiated and when you want to force subclass to override methods you will use them.



Abstraction: Hiding the implementation & showing only the necessary details to the user.

Abstraction can be achieved in 2 ways:

1. Abstract class: partial abstraction
2. Interface: 100% abstraction

TestInterfaceDemo.java

