Software Requirements

* JDK 17 or later
* Eclipse IDE or command prompt and notepad

Java: Platform independent and object oriented programming language

Object Oriented programming language has 2 building blocks

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

class Customer {   
 // properties & behaviors  
 // properties – variables  
 // behaviors - methods  
}  
Customer c1 = new Customer();

JDK: Java Development Kit

JRE: Java Runtime Environment

JVM: Java Virtual Machine

Java Fundamentals: Base for Java programmers

Datatypes

Operators

Conditions

Loops

Arrays

Datatypes: These are set of keywords which are used to create variables that can store some value, there are two types

1. primitive types – size is fixed
2. derived types (combination of primitives) – size varies

Primitive types

|  |  |
| --- | --- |
| Type | Size in bytes |
| byte | 1 (-128 to +127) |
| short | 2 |
| int | 4 |
| long | 8 |
| float | 4 |
| double | 8 |
| char | 2 (‘M’, ‘F’) |
| boolean | 1 (true, false) |

Derived types

1. class
2. arrays

Operators

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

Conditional Statements

1. simple if
2. if & else
3. if, else if, else if …. else
4. switch

if(conditions) {   
  
}  
else if (conditions) {   
  
}  
else {  
  
}

Loops: It is to iterate the statements until some condition is true, there are 3 types of loops

1. for
2. while
3. do while

Activity:

1. Enter 3 digits numbers & print their digit with words  
   ex: 890 must print Eight Nine Zero
2. Enter 3 digits and add the highest digit and lowest digit and display the result  
   ex: 759 must add 9 + 5 and print 14
3. Create an array of some elements and display the maximum, minimum & sum of the array, using only one loop perform all the operations  
   ex: items = {7, 1, -1, 9, 10, 15, 8}, then maximum = 15, minimum = -1, sum = 49

Classes & Objects

Inside class you write variables, methods & constructors

variables: They store data of an object like id, name, salary, phone, email and etc

methods: They will have logics like display(), debit(), credit() and etc

constructors: They will have initialization logics

Command line arguments: It is used to pass input to the program before launching the program

java HelloWorld arg1 arg2 arg3

arg1, arg2, arg3 are stored in args of main method

Day 2 Agenda

Overloading

Static

OOPS concepts

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

Naming Conventions

1. Class Names: Begin with uppercase & follow camel case when there are more than one word ex: HelloWorld, StringBuffer, RuntimeException
2. Variables & Methods: Begin with lower case & follow camel case ex: nextInt(), nextFloat(), charAt()
3. Project Names(Optional): You can begin with lower case and separate by hyphen when there are more than one word

Constructor Overloading:

When multiple constructors are written in the class, it will be constructor overloading

class Person {   
 // name, gender, email & phone  
 Person(String name, String gender) { }  
 Person(String name, String gender, String email) { }   
 Person(String name, String gender, String email, long phone) { }  
 Person(String name, String gender, long phone) { }  
}

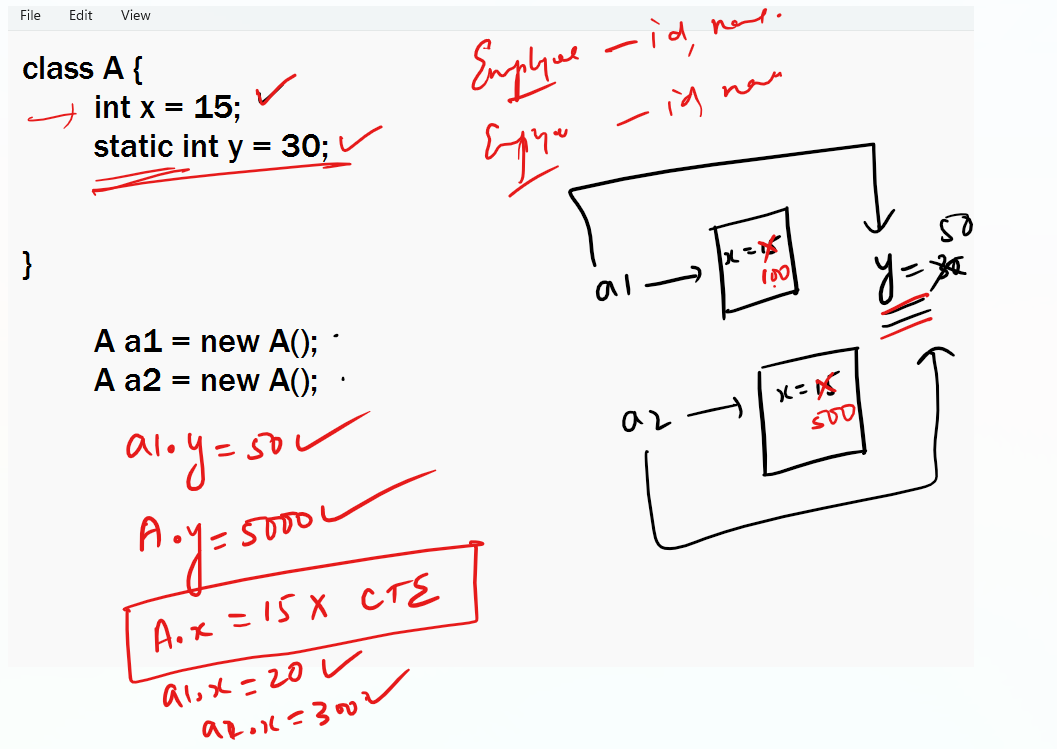
Method Overloading

A class with multiple methods having same name but different signature(parameters, parameters type)

class Calculator {   
 int add(int x, int y) { }   
 float add(float x, float y) { }  
 String add(String x, String y) { }   
}

Static members:

Static variables & static methods can be accessed without creating objects

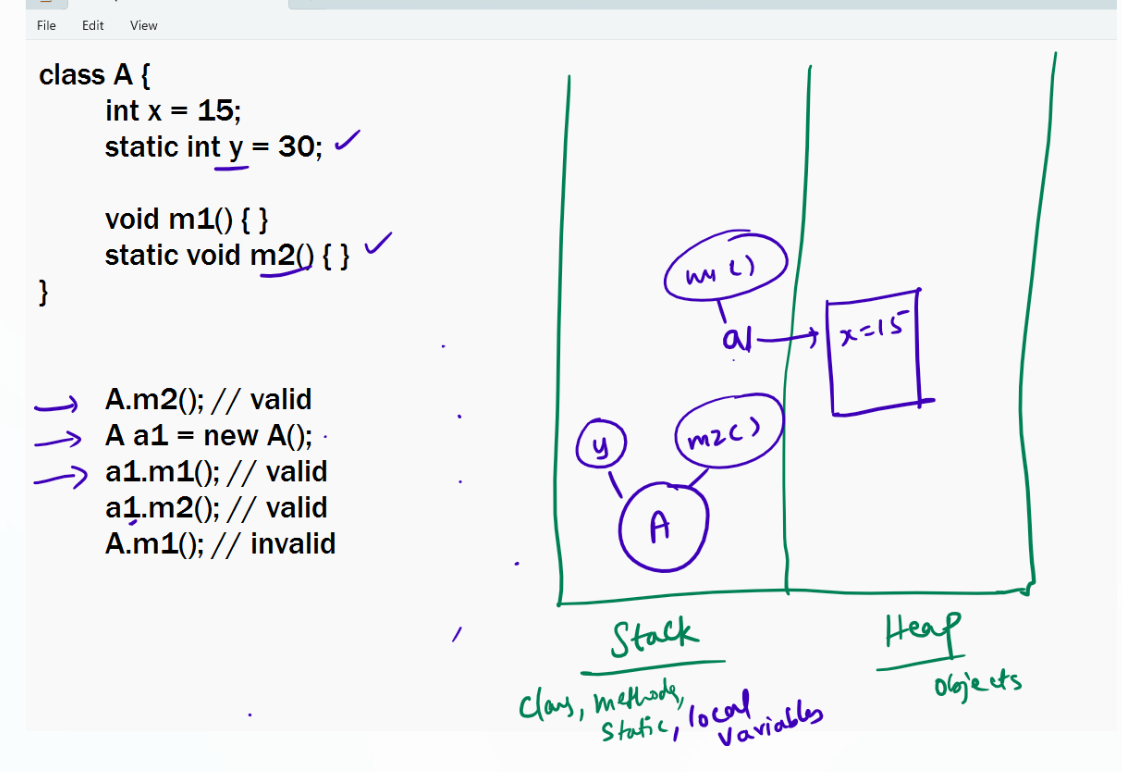


You can access static members using class name or reference variable,

instance variable: every object will have their own copy of instance variable, ex: id, name, salary, phone can be instance variable of Employee

static variable: all the objects share single copy of static variable, ex: company name of Employee

Memory allocation of static & instance members



OOPs principles

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

Encapsulation: Hiding the data(variables) and accessing them only through public methods, so that you will have control over the data, it is achieved by making variables private & methods public

public methods: These are setters & getters which are used to modify the data & read the data

Inheritance: It is used to acquire properties & behaviors of a class from another class so that you don’t have to rewrite properties & behaviors

In Java you use extends keyword to inherit

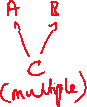
Inheritance types



1. Single level

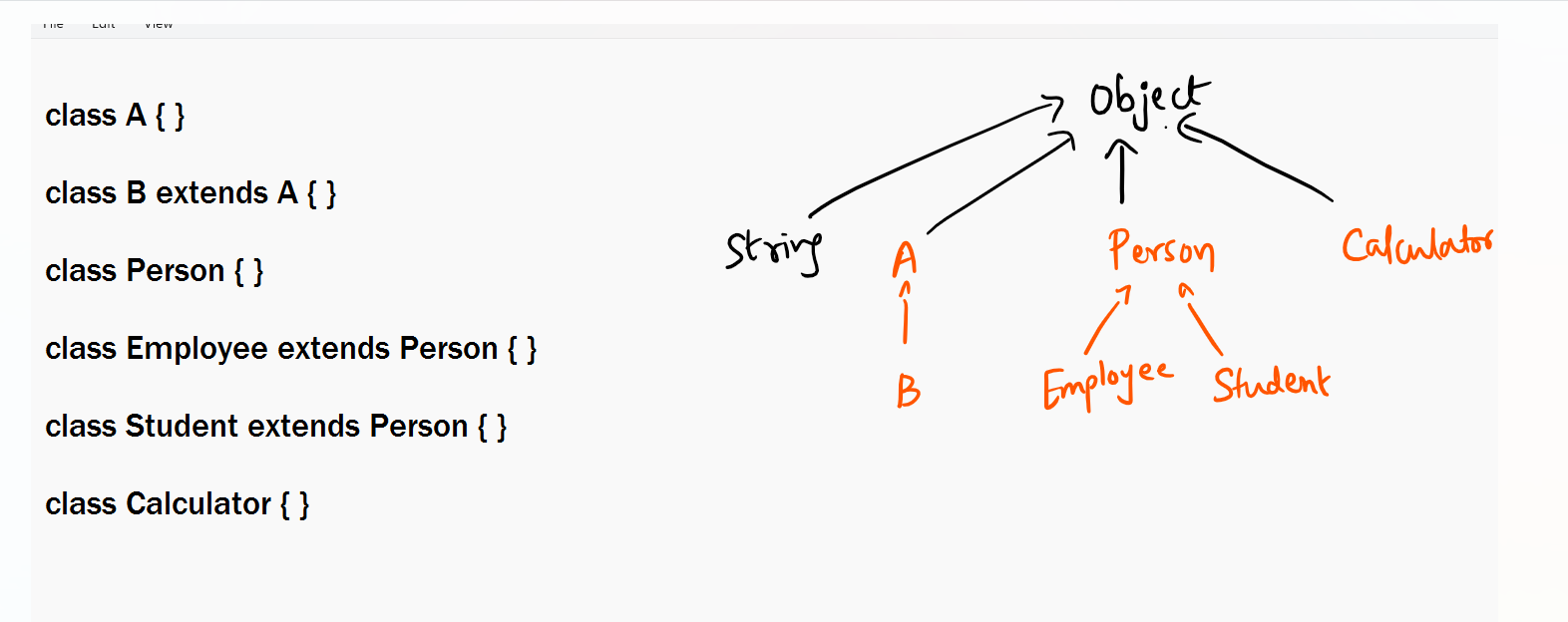


1. Multi level



1. Hierarchical
2. Multiple (No supported in java classes, but supported in java interface)

Object: It is the root class in Java, every class either directly or indirectly inherits Object



Polymorphism: Poly means many, morphism means forms, it allows a method to have many forms

ex: power button acts like on / off

Similarly in OOPs you can create a method with many forms, this can be achieved in 2 ways

1. Method Overloading: Same method name but different signature in the same class
2. Method Overriding: Same method & same signature but different logics in the subclass

Activity:

1. Create Student & Customer class with some properties & provide getters, setters, constructors as per the properties, both of them must inherit Person, override display method in both the classes & print their respective details, in main method create object of Student & Customer and pass them to the test method & observe how the display() method of test method prints student & customer details.

Access specifiers: These are used to specify the visibility for class, variables, methods & constructors, there are 4 access specifiers in Java

1. private: visible within the class
2. public: visible to all
3. protected: visible within the package & outside the package only to the subclass
4. doesn’t have any keyword – its called default scope: visible only within the package

final keyword: It is used on a class, variable and method

final variable: It is constant you can’t modify

ex: final int x = 20; // you can’t modify x

final method: Overriding is restricted

ex: final void display() { } // you can’t override

final class: You can’t make a subclass

Abstraction: Hiding the implementation and showing only the necessary details to the end user, it is achieved using interfaces & abstract classes

interfaces: It provides only method signatures without method body

interface X {   
 void store(); // abstract methods  
 void delete(); // abstract methods   
}

class Imp1 implements X {   
 // they must override all the abstract methods  
}

class Imp2 implements X {   
 // this must override all the abstract methods  
}

Advantages of abstraction

1. Client code doesn’t need to be modified if the implementation changes
2. You can hide the methods at the client side by restricting him to access only few methods instead of accessing all the methods