International study centre

AN INTRODUCTION TO OBJECT-ORIENTATION AND THE JAVA PROGRAMMING LANGUAGE

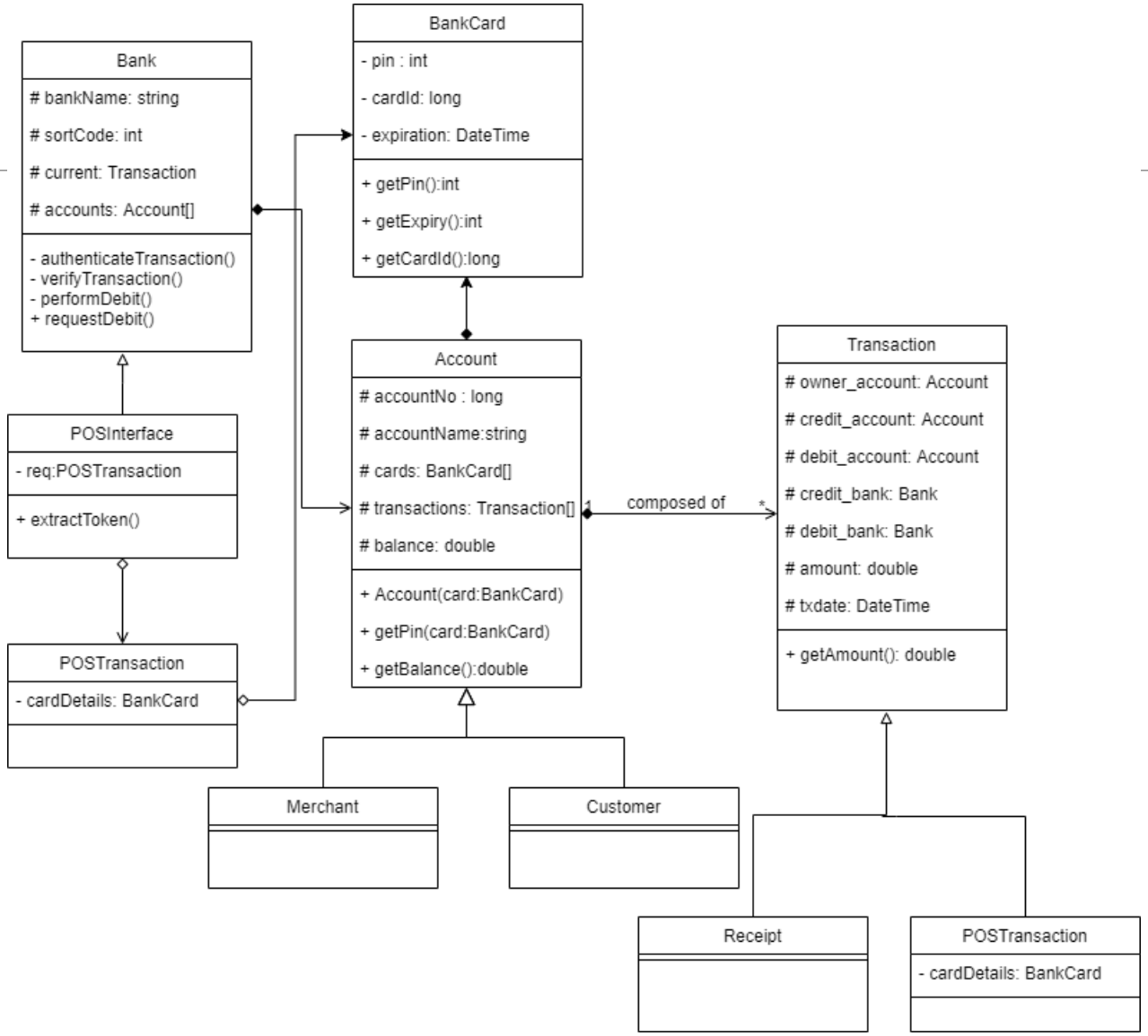
ADVANCED OOP IN JAVA

- Name : John Alamina
- Email : john.alamina@hud.ac.uk

Outline

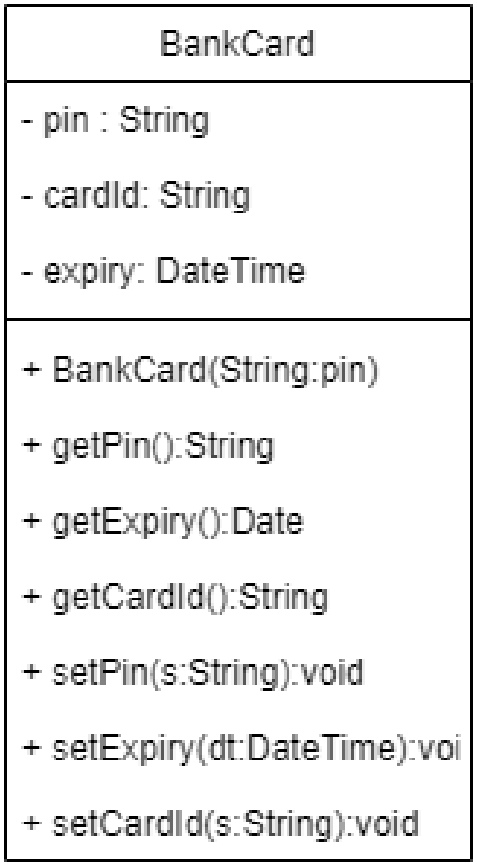
- Getters and setters
- Inheritance
- Polymorphism
- Instantiation vs Inheritance
- Composition vs Inheritance
- Interface vs Inheritance
- Abstract class vs interface vs inheritance
- Inner classes
- Java packages
- Java library and standard classes

POS Class diagram Example



Getters and setters

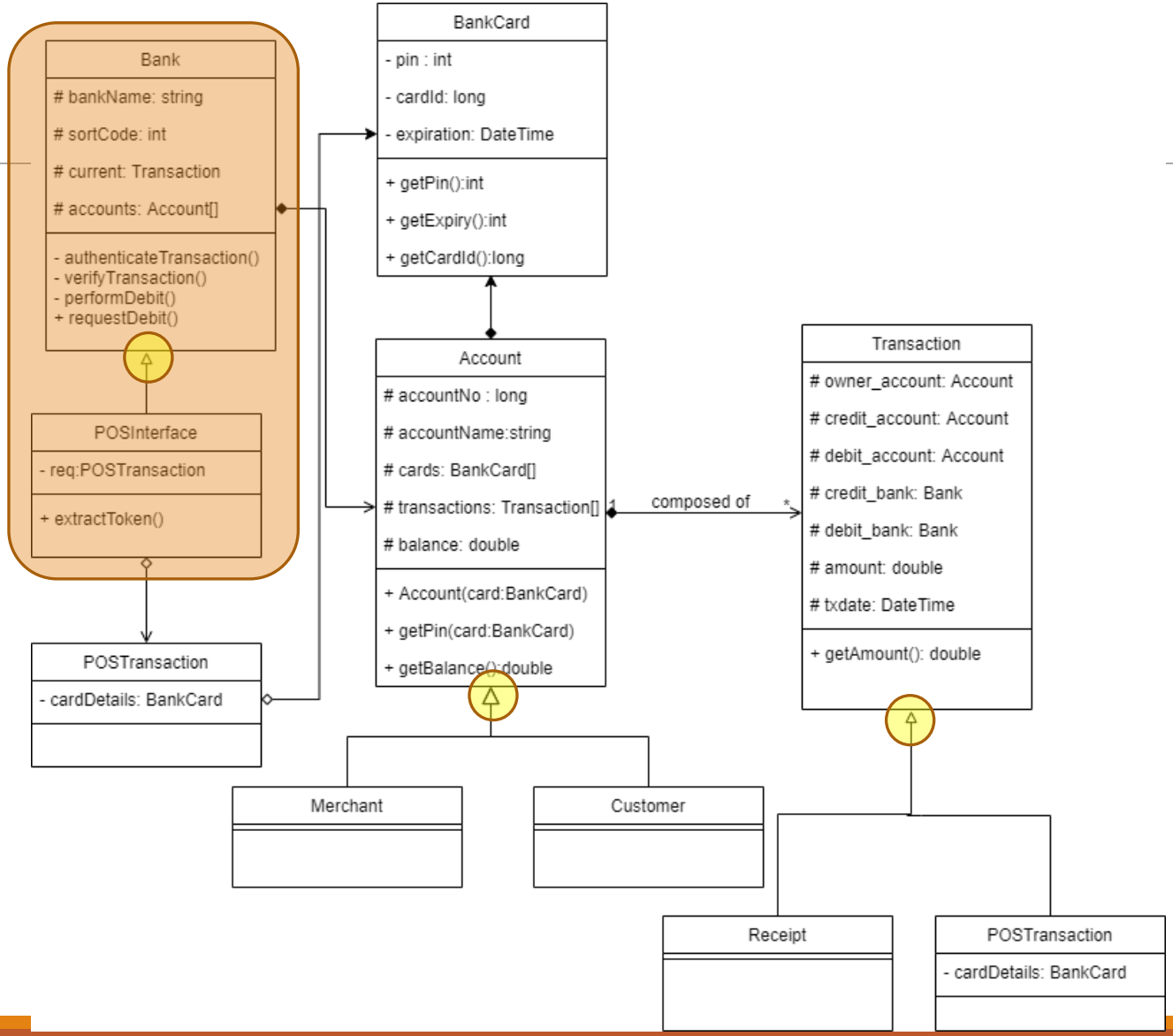
- Getters and setters are a means of externally accessing private/protected members externally




```
public class BankCard {
   String pin="";
   String cardid="";
   Date expiry;

   public void setPin(String p){
      this.pin=p;
   }
   public String getPin() {
      return pin;
   }

   public Date getExpiry() {
      return expiry;
   }

   public void setExpiry(Date expiry) {
      this.expiry = expiry;
   }

   public String getCardid() {
      return cardid;
   }

   public void setCardid(String cardid) {
      this.cardid = cardid;
   }
}
```

Any Questions?

POS Class diagram - Inheritance



Any Questions?

Polymorphism

- Polymorphism is related to how functions or methods are implemented.
- There are two types of Polymorphism
- Function polymorphism also known as function overloading or method overloading
- Object polymorphism also known as method overriding.
- Note that method overloading occurs within a single class but overriding is between two or more classes having an inheritance relationship (object polymorphism)

Any Questions?

Method overloading - Example

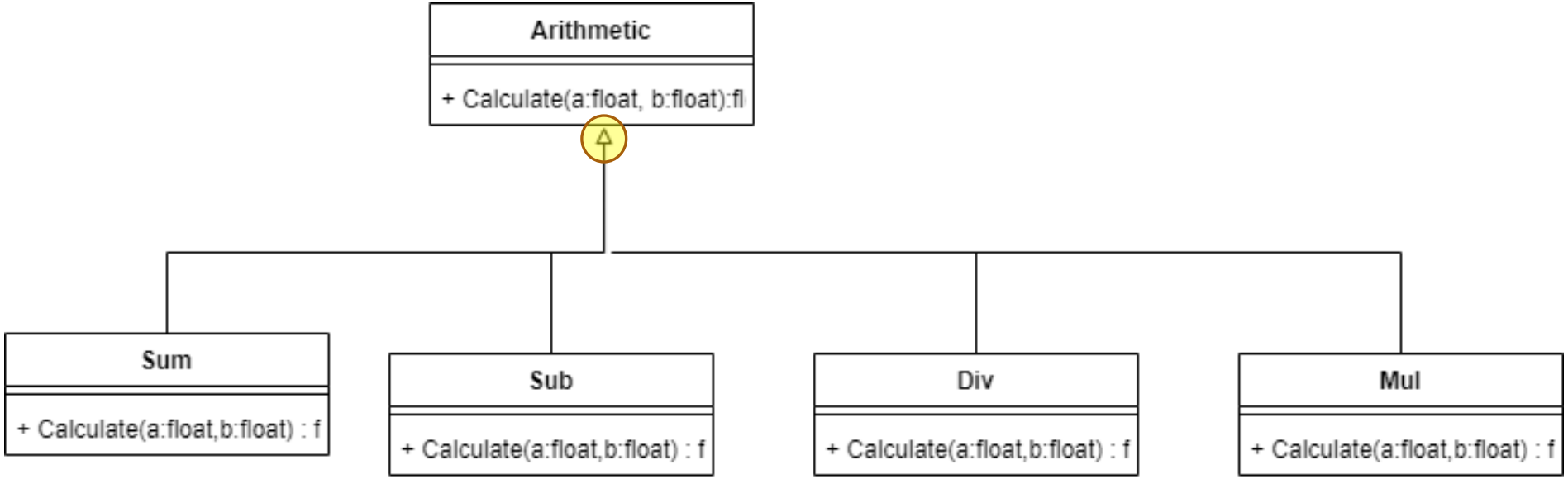
- Note that for method overloading to work, the input parameter types or argument types must be different in other words, the method signature or prototype must be different for each overloaded method.
- Voltage can be obtained from various parameters. It can be obtained by multiplying the current, I by the resistance, R. Also voltage can be obtained by dividing charge, Q by capacitance, C. When Q can also be derived from current, I, multiplied by the time, t. Thus

$$V=IR=It/C$$

- We can therefore write two overloaded methods of getVoltage(). The first overloaded method having two double parameters and the other having three double parameters
- getVoltage(double I, double R);
- getVoltage(double I, double t, double C);

Any Questions?

Method overriding - Example

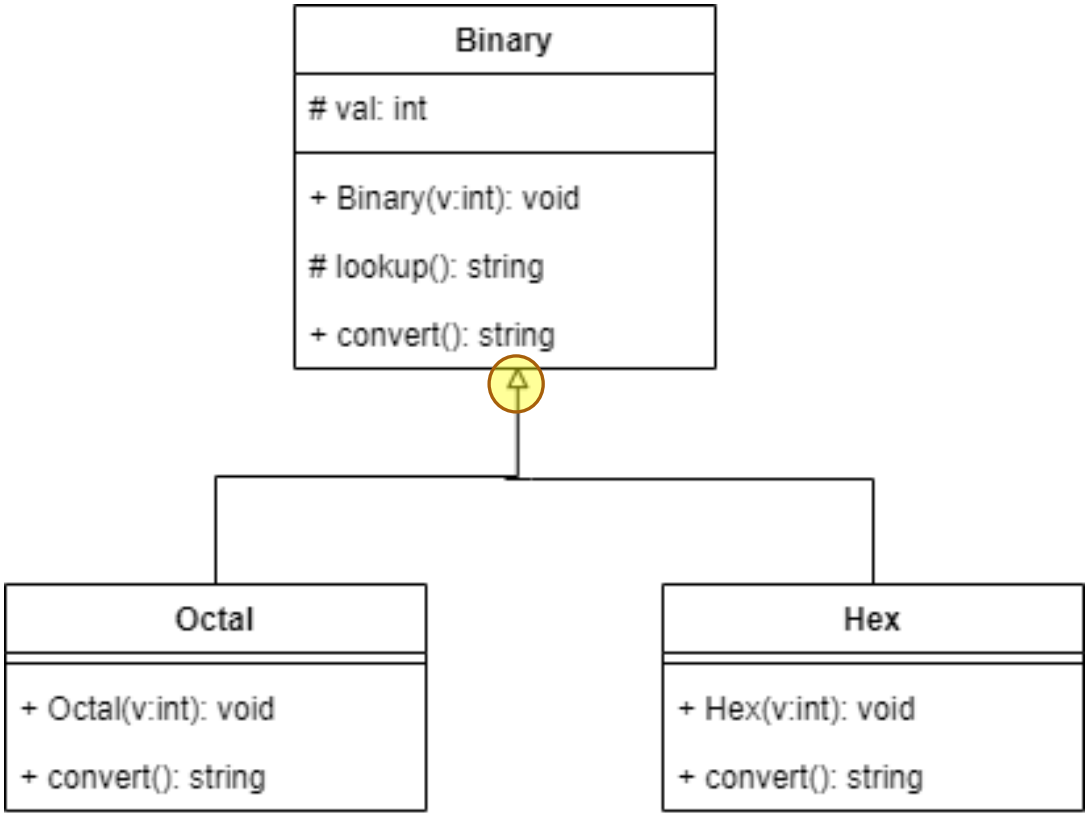


Method Overriding - Example

```
public class Div extends Arithmetic{
  public double calculate(double a, double b){
    return a/b;
  };

}
public class Mul extends Arithmetic{
  public double calculate(double a, double b){
    return a*b;
  };
}
public class Sum extends Arithmetic{
  public double calculate(double a, double b){
    return a+b;
  };
}
public class Sub extends Arithmetic{
  public double calculate(double a, double b){
    return a-b;
  };
}

public class Arithmetic{
  public double calculate(double a, double b){r
eturn 0;}

 }
```

```
public class ArithMain{
  public static void main(String ar[]){
    double x,y;
    System.out.print("Enter the first value: ");
    x=TextIO.getDouble();
    System.out.print("Enter the second value: ");
    y=TextIO.getDouble();
    System.out.print("Enter the operation (+,-,/,x) ");
    char o=TextIO.getChar();
    Arithmetic op;
    switch(o){
      case '+':
        op=new Sum();
        break;
      case '-':
        op=new Sub();
        break;
      case 'x':
        op=new Mul();
        break;
      case '/':
        op=new Div();
        break;
      default:
        op=new Arithmetic();
        System.out.println("Invalid operation selected")
;
      break;
    }
    double result=op.calculate(x,y);
    System.out.println(String.format("%3.1f %s %3.1f = %
3.2f",x,o,y,result));
  }
}
```

Method overriding - Example

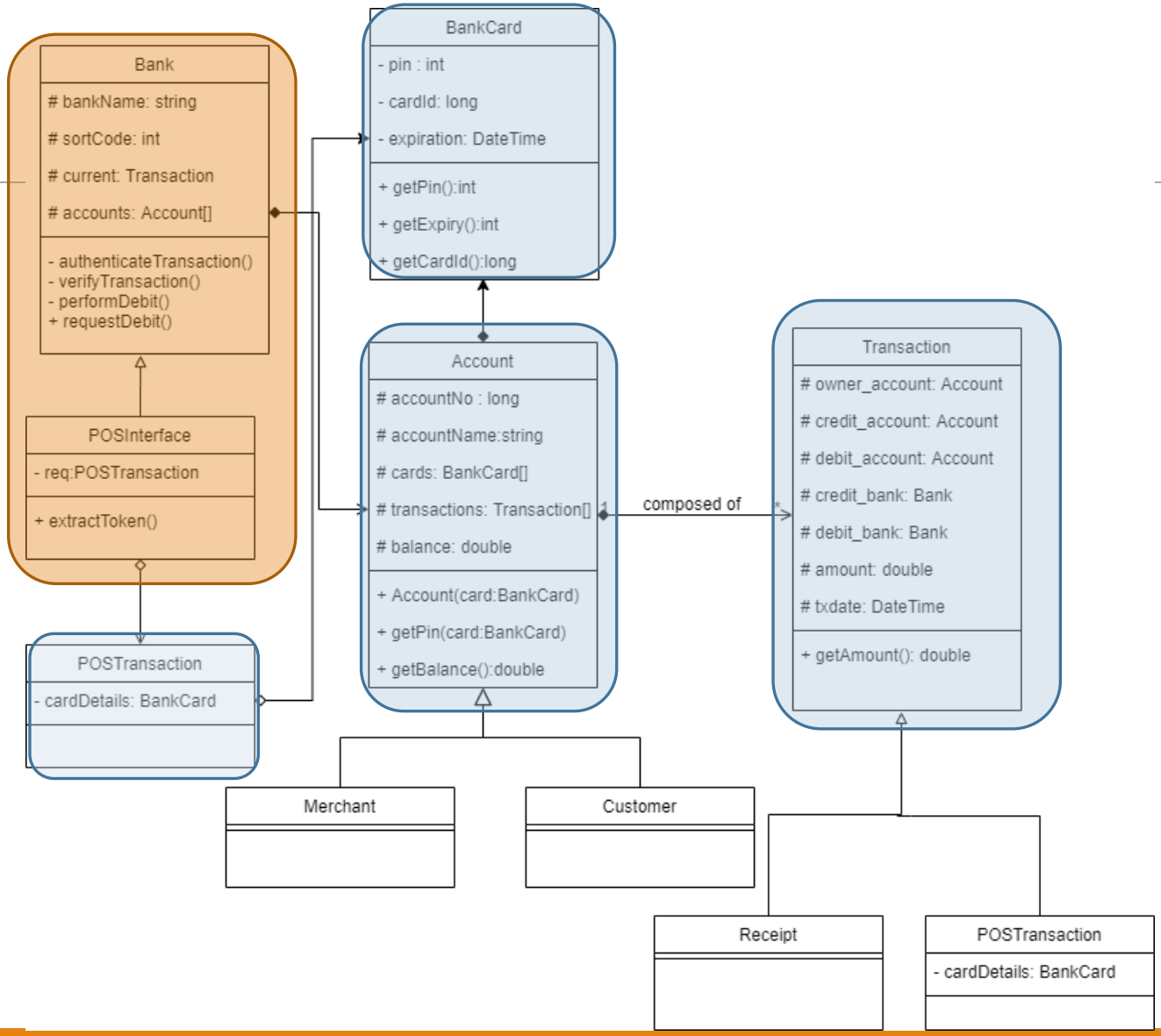


Any Questions?

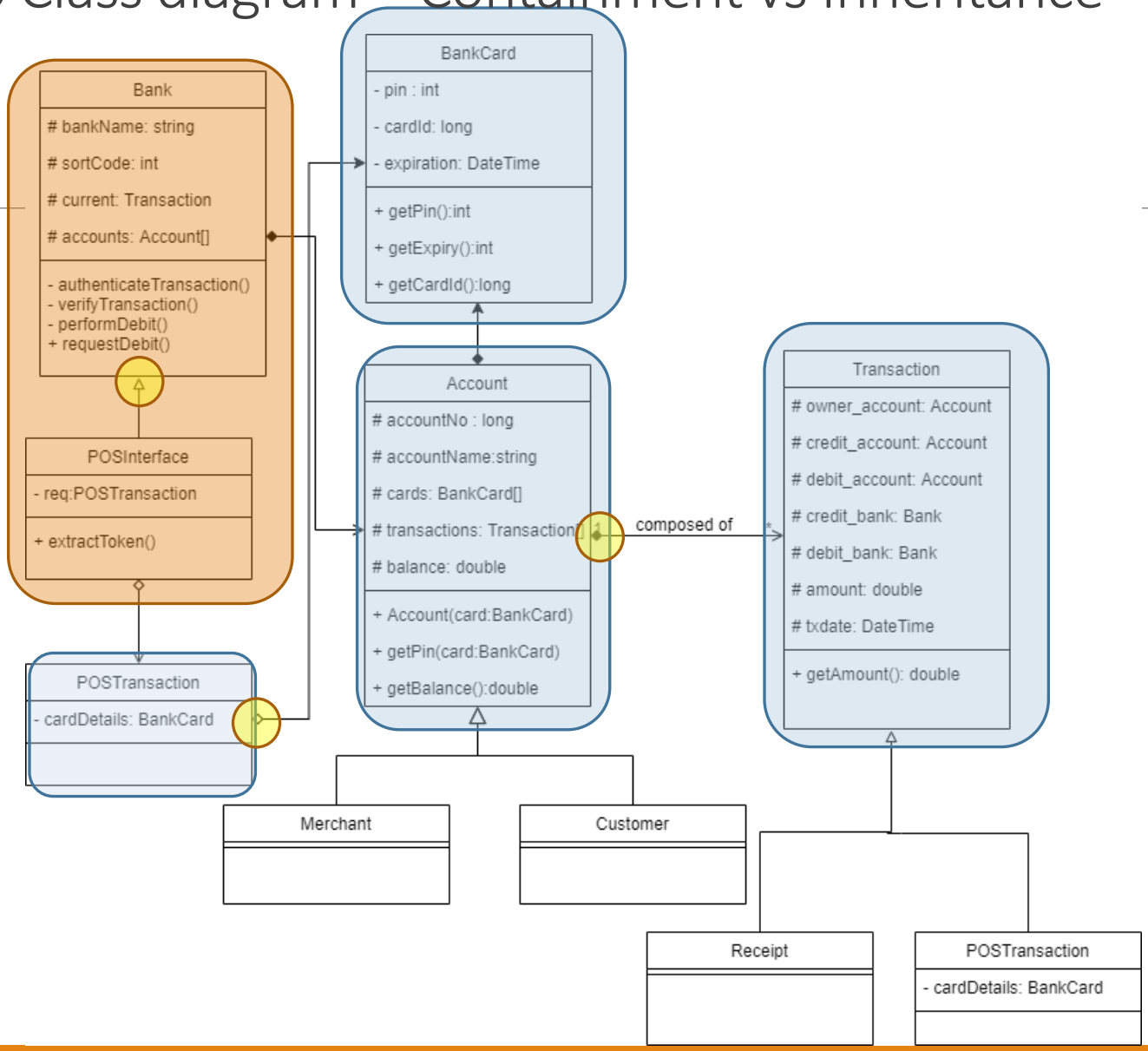
Instantiation vs Inheritance

- Instantiation is the process of creating a new Object from a class.
- e.g Arithmetic op=new Arithmetic();
- The new keyword along with the class constructor is used to create a new object having the state as dictated by the constructor.
- A constructor is a special method that doesn't have a return type and must have the same name as the name of the class and is used to assign values to an object members, giving it a state.
- Classes cannot be used directly. The only way to access members of a class is from the object of the class unless that member was declare as static.
- Inheritance however is done during class definition where one class extends from another e.g.
- public class Div extends Arithmetic{}

POS Class diagram – Containment vs Inheritance

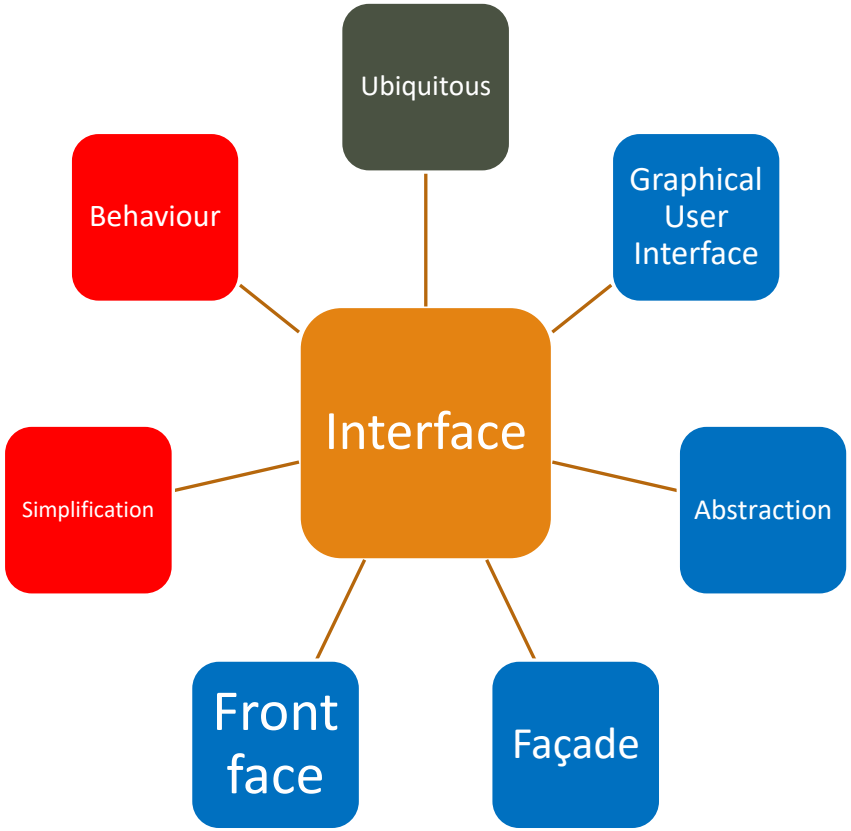


POS Class diagram – Containment vs Inheritance



Any Questions?

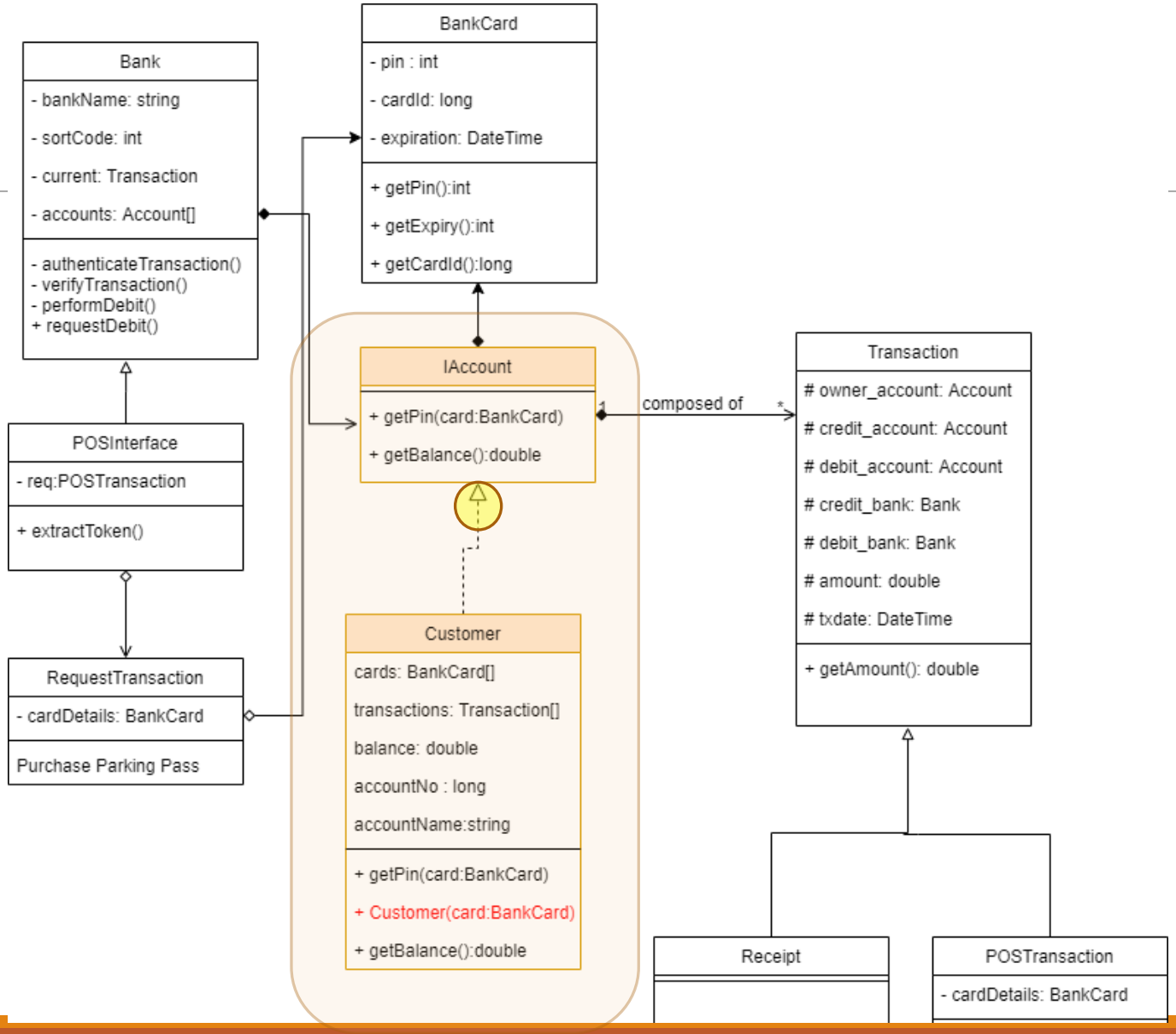
Interface vs Inheritance – Why so many faces?



Interface vs Inheritance

- An interface is a special type of inheritance
- An interface is a contract that every inherited class must keep by ensuring every method declared in the parent interface is overridden (object polymorphism)
- An interface cannot implement any of its methods and doesn't have any data members.
- In Java interfaces are usually used to achieve a concept known as multiple inheritance in C++, where a class can inherit from more than one superclass.
- Interfaces can be used to achieve loose coupling while maintaining high cohesion

Interface vs Inheritance

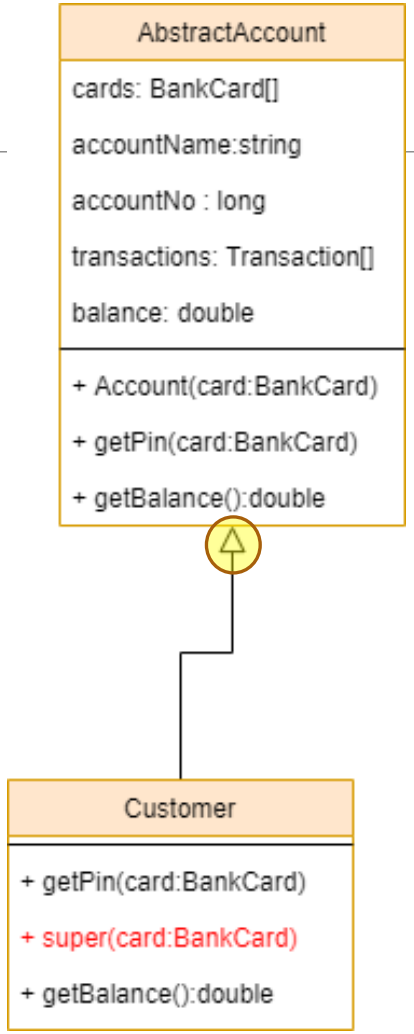


Any Questions?

Abstract class vs Interface vs Inheritance

- There are limitations associated with an interface which include:
  1. An interface does not have data members only methods.
  2. All the methods declared within an interface must be implemented.
- An abstract class is a go between a super class and an interface because it allows
  1. certain methods that must be overridden and others that are optionally overridden.
  2. data members which can be private, public, protected etc.
- Methods within an abstract class that must be overridden are called abstract methods.

Abstract class diagram

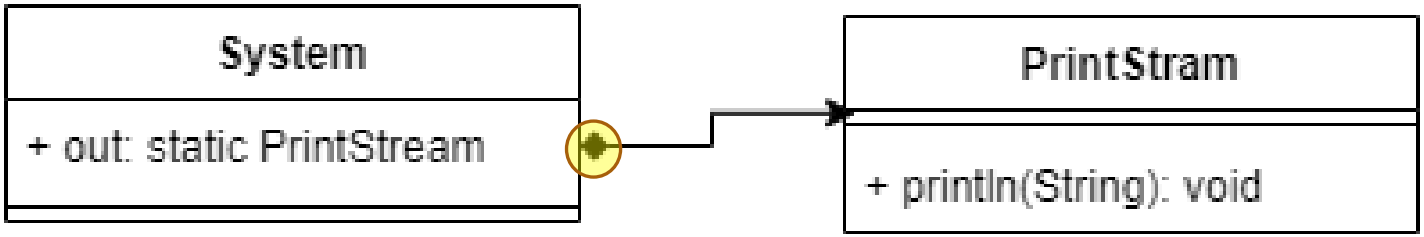


Any Questions?

Inner Classes

- Inner classes are used where there is close relationship between two classes and the inner class is only accessed by the containing class.
- An example of an inner class is the System.out class we have been using all this while.
- System is the outer class and out is the inner class.
- System.out class has several methods most notably print() and println() which print strings out to the console.
- System also has a System.in inner class used for input we will see in this in action later in this boot camp.
- Read more about inner classes here.

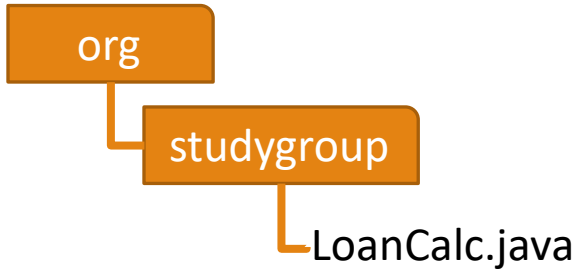
Inner-class representation



Any Questions?

Java Packages

- OOP helps to organise your code into reusable logical structures known as classes.
- A large number of classes have been written by the creator of the java language and arranged into logical hierarchies known as packages.
- In practice as every java class represents a compilation unit or file, every package represents an actual sub-folder within your operating system file system.
- thus org.studygroup.LoanCalc class has the following hierarchy



Java Packages

- Packages are imported using the import keyword and declared using the package keyword.

e.g.

```
package addbook; //references AddressBook class inside addbook folder
import java.util.ArrayList; //imports ArrayList class from java.util package

class AddressBook{
 private ArrayList addressBook =new ArrayList();  //using the imported class
}
```

Any Questions?

The Java Standard Edition Library

- In C++ there are a lot of functions and objects a programmer can import from the C++ standard library
- Java has a rich library which consists of packages and classes.
- You can study this in-depth by using the link provided at the end of the slides.
- Two common packages referred to in this bootcamp is the java.lang package and the java.util package.
- The java.lang package is automatically imported by the java compiler and contains classes like the System class we use for console output and the Math class.
- The java.util classes have special utility classes including the Collection package we consider in a later lesson

Common Java Language (java.lang) Package Classes

- String
- Math
- Array
- Integer
- Double
- Float
- Character
- Byte
- StringBuffer
- System
- Number
- Object
- StringBuilder
- Class
- Process
- Boolean
- Short
- Long

Further Reading

- String builder vs String buffer

Any Questions?

Exercise 1

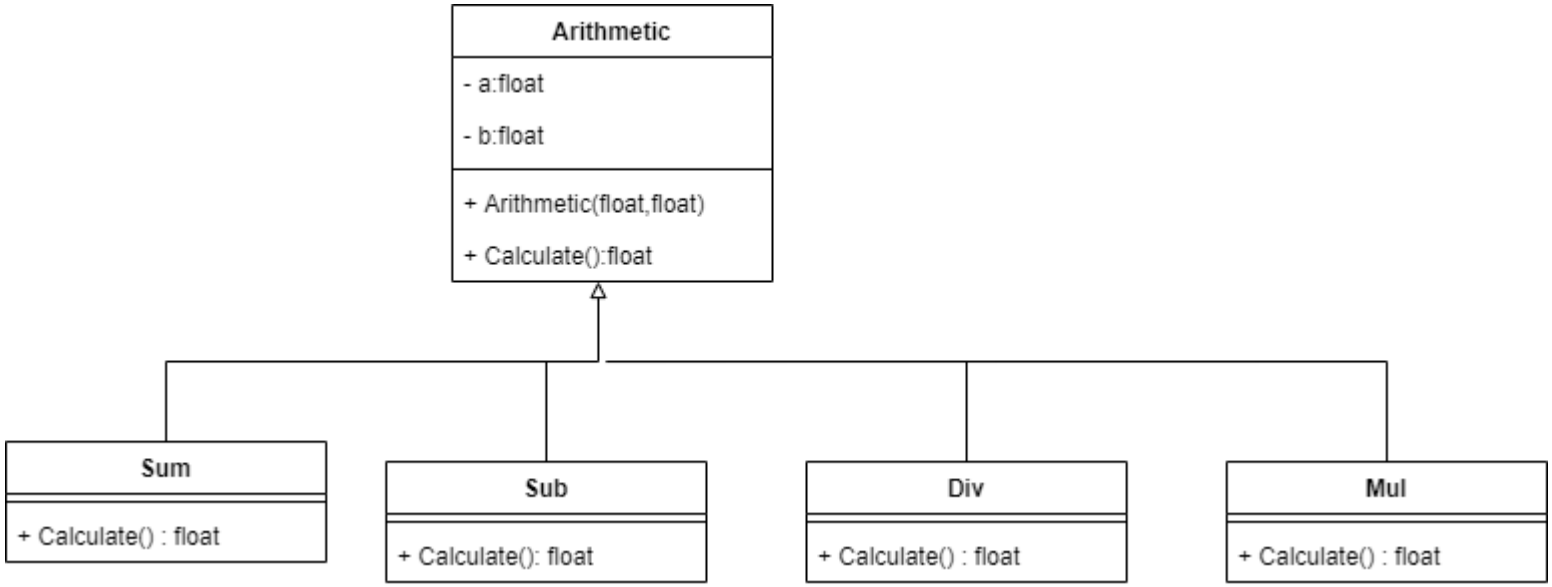
- Voltage can be obtained from various parameters.

$$V=IR=It/C=IpL/A$$

- Where
  - I = Current (amperes)
  - R = Voltage (volts)
  - t = time (seconds)
  - C = Capacitance (farads)
  - p = Resistivity (Ohm-meter) of copper=$1.678 \times 10^{-8} \Omega m$
  - L = Length
  - A = cross sectional area
- Write a program to calculate voltages for the following program.
  1. A Current of 12A across a Capacitor of 0.5F for 5seconds.
  2. Copper length of 1m and cross-sectional area of 0.001m and a current of 10A.

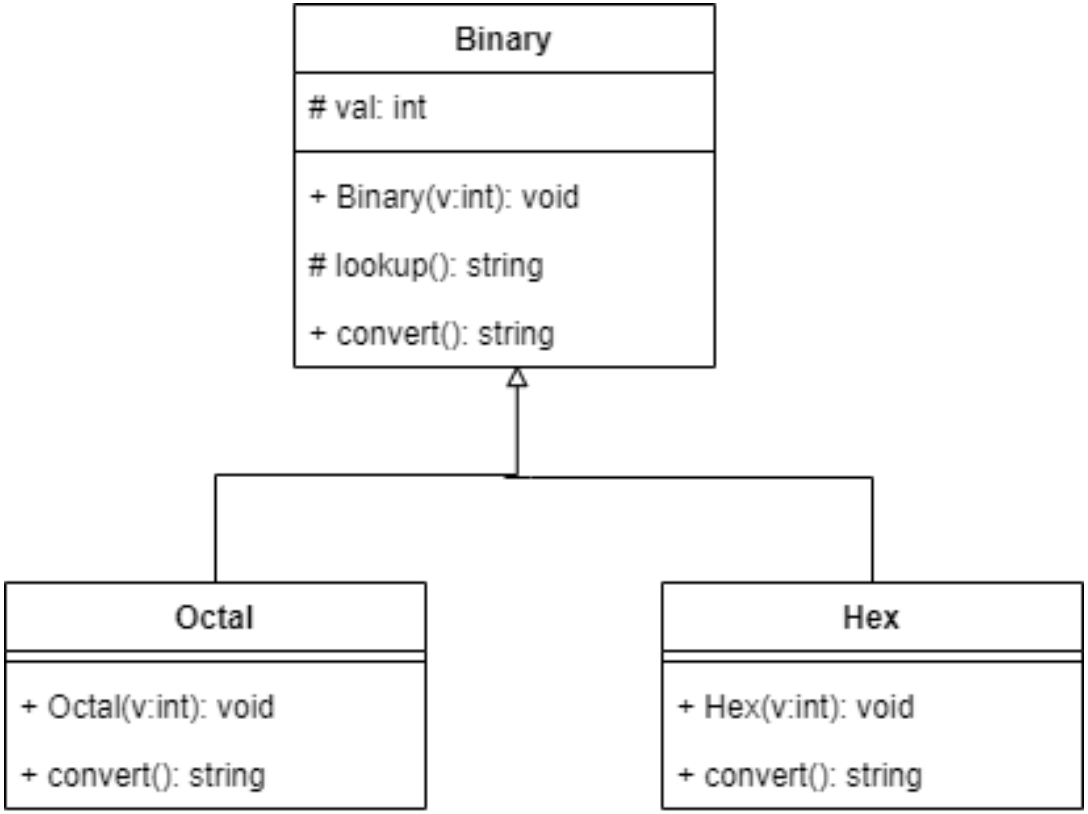
Exercise 2

Implement the class diagram below using object polymorphism
Note that this is the second implementation done earlier



Exercise 3 (advanced)

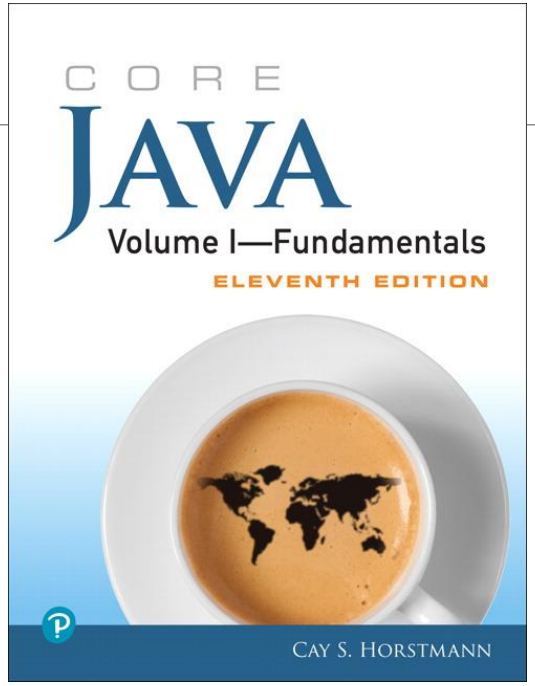
Implement the class diagram below using object polymorphism. Starter C++ code can be found here

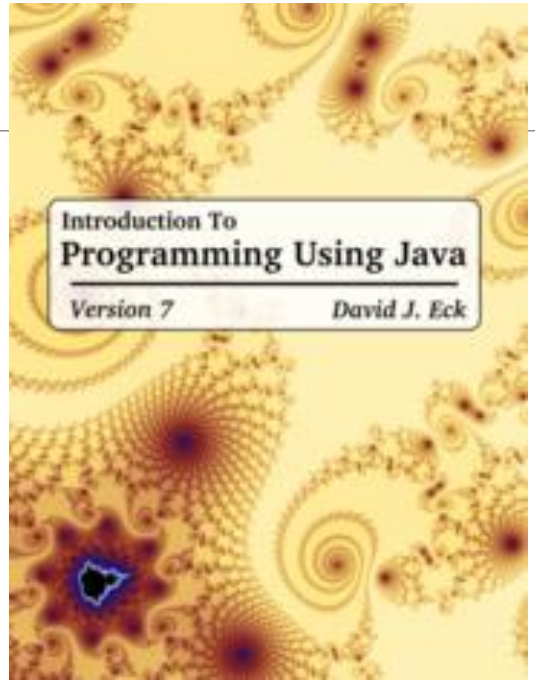


Exercise 4

- Write a program using static cosine function in java.lang.Math and method overloading to find the parameters of a triangle (3 sides and 3 angles) given either 2 sides and one angle or two angles and one side. Assume the angles are always whole numbers and while sides are real numbers.

Supplementary material





- The Java Tutorial
- Java API documentation
- Link to today's Session screencast
- Link to John's Group Padlet
- Link to Kelly's Group Padlet