City University

Object oriented programming

Lecture 3

Supta Richard Philip

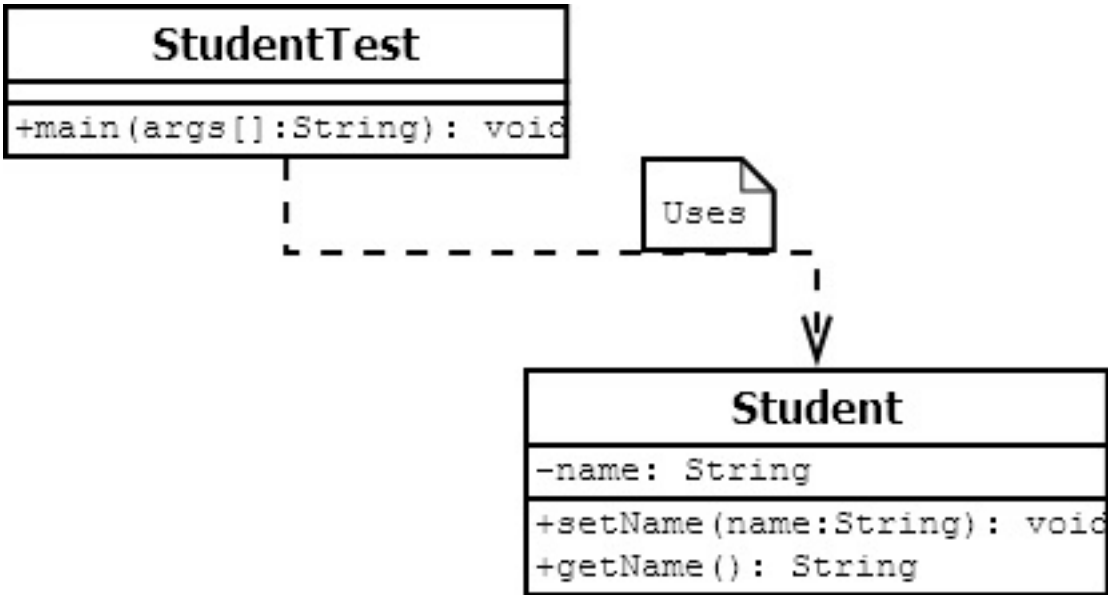
supta.philip@gmail.com

The main goal of OOP is to bind data and code(methods) togather.

OOP provide few concepts

1. Encapsulation

- Encapsulation in Java is a process of wrapping code and data together into a single unit.
- create a fully encapsulated class in Java by making private all the data members of the class.
- use setter and getter methods to set and get the data in it.
- The Java Bean class or POJO class is the example of a fully encapsulated class.
- Class Diagram of Student



Student.java

```
package org.cityU.Encapsulation;
public class Student {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

StudentTest.java

```
package org.cityU.Encapsulation;

public class StudentTest {
```

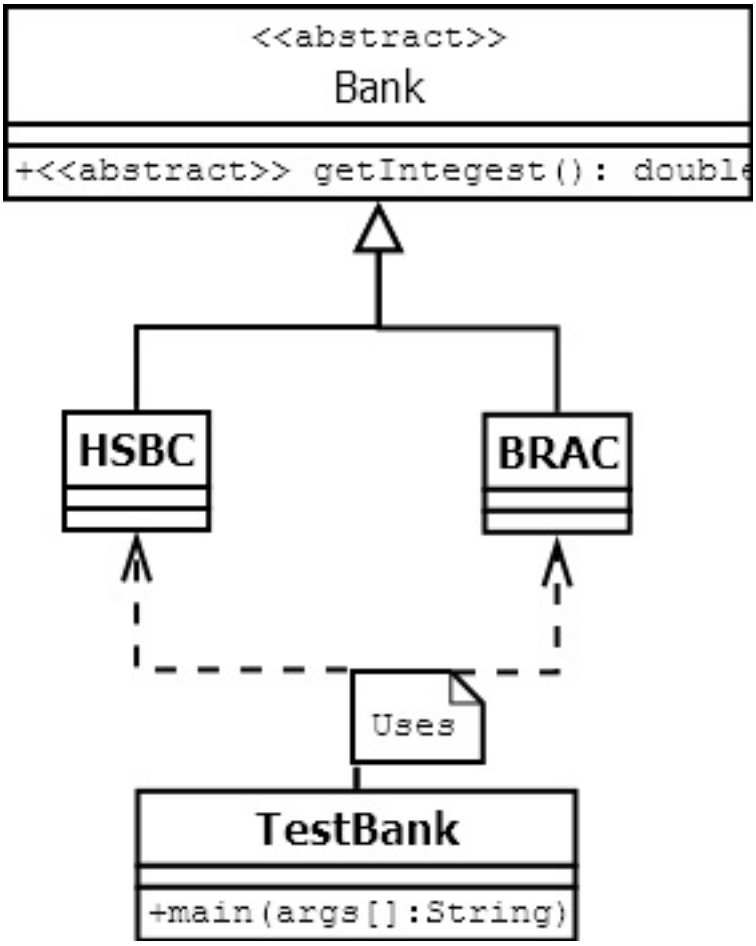
```
    public static void main(String[] args) {

        Student s = new Student();
        s.setName("Richard");

        System.out.println(s.getName());
    }
}
```

2. Abstraction

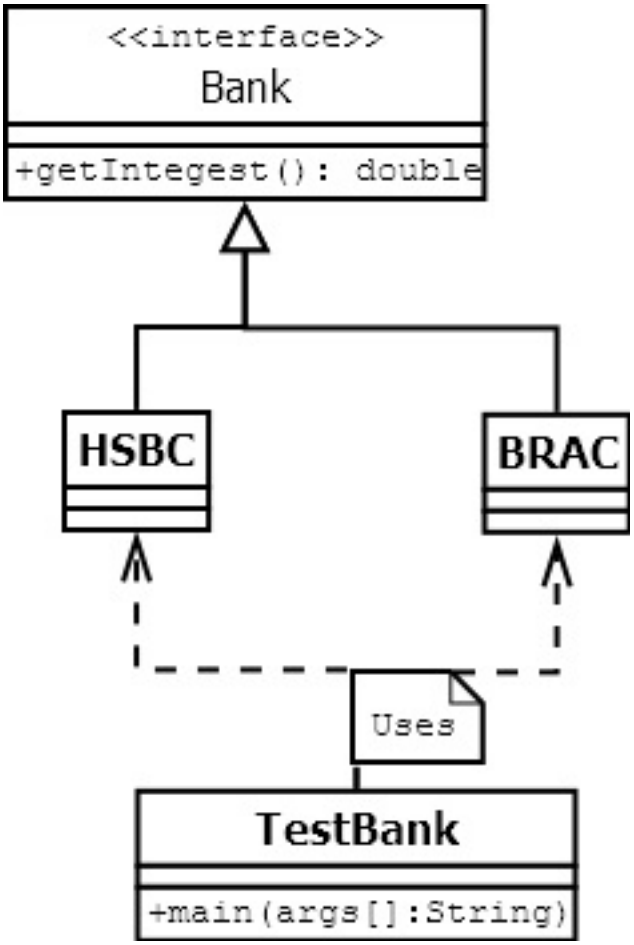
- Abstraction is a process of hiding the implementation details and showing only functionality to the user.
- There are two ways to achieve abstraction in java
    - Abstract class (0 to 100%)
    - Interface (100%)




```
abstract class Bank {
    abstract int getRateOfInterest();
}
class HSBC extends Bank {
    int getRateOfInterest() {
        return 7;
    }
}
class BRAC extends Bank {
    int getRateOfInterest() {
        return 8;
    }
}
```

```
class TestBank {
    public static void main(String args[]) {
        Bank b;
        b = new HSBC();
        System.out.println("Rate of Interest is: " + b.get
RateOfInterest() + " %");
        b = new BRAC();
        System.out.println("Rate of Interest is: " + b.get
RateOfInterest() + " %");
    }
}
```

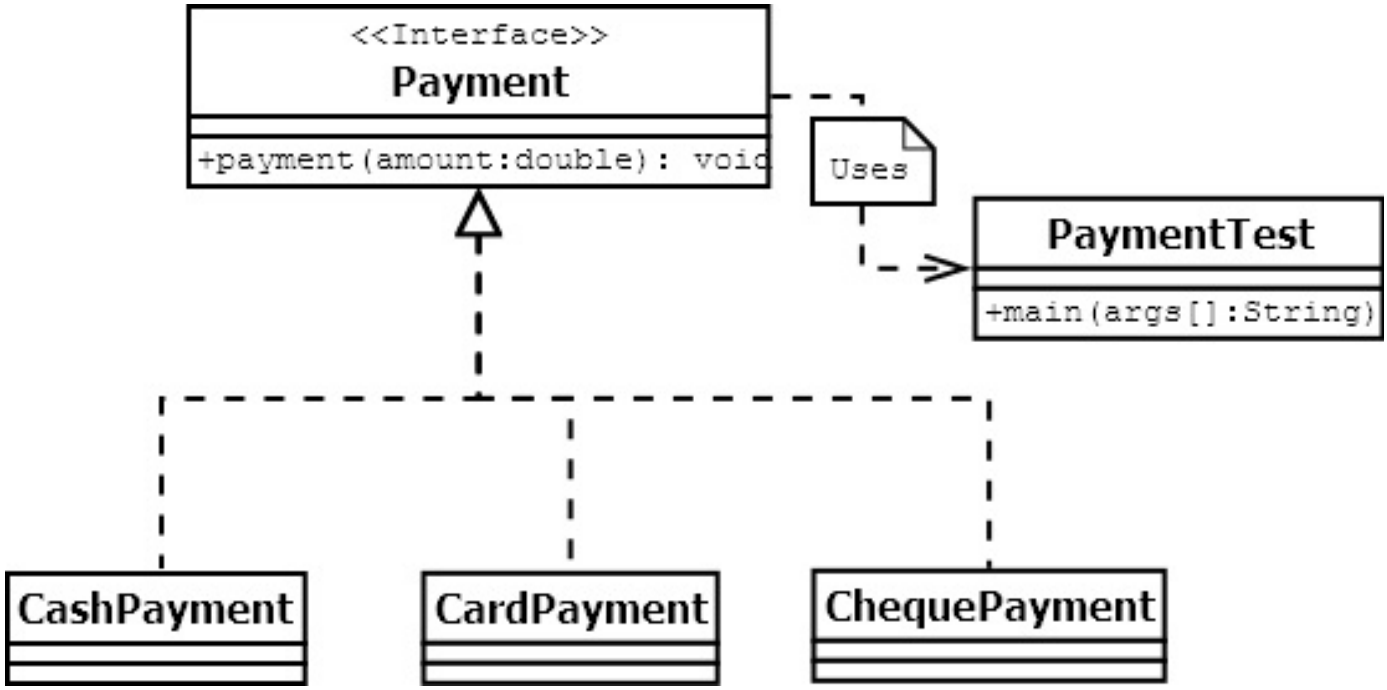
Same examples you can solve with interface.



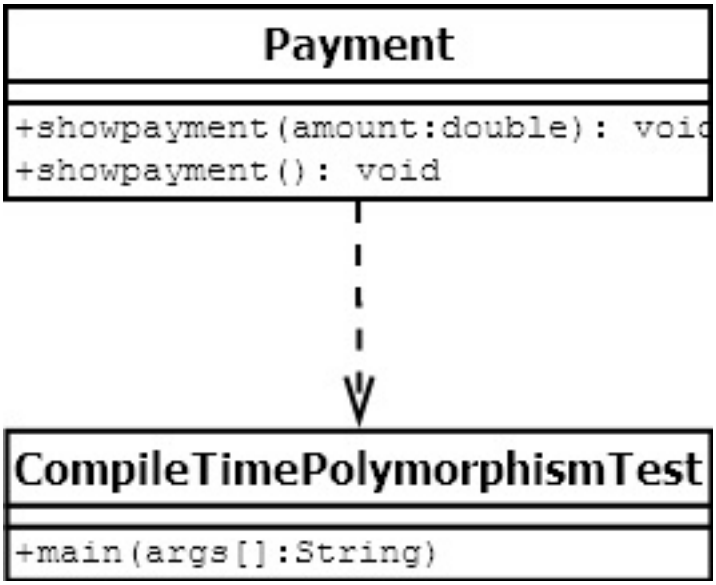
3. Polymorphism

- Polymorphism in Java is a concept by which we can perform a single action in different ways.
- There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism.

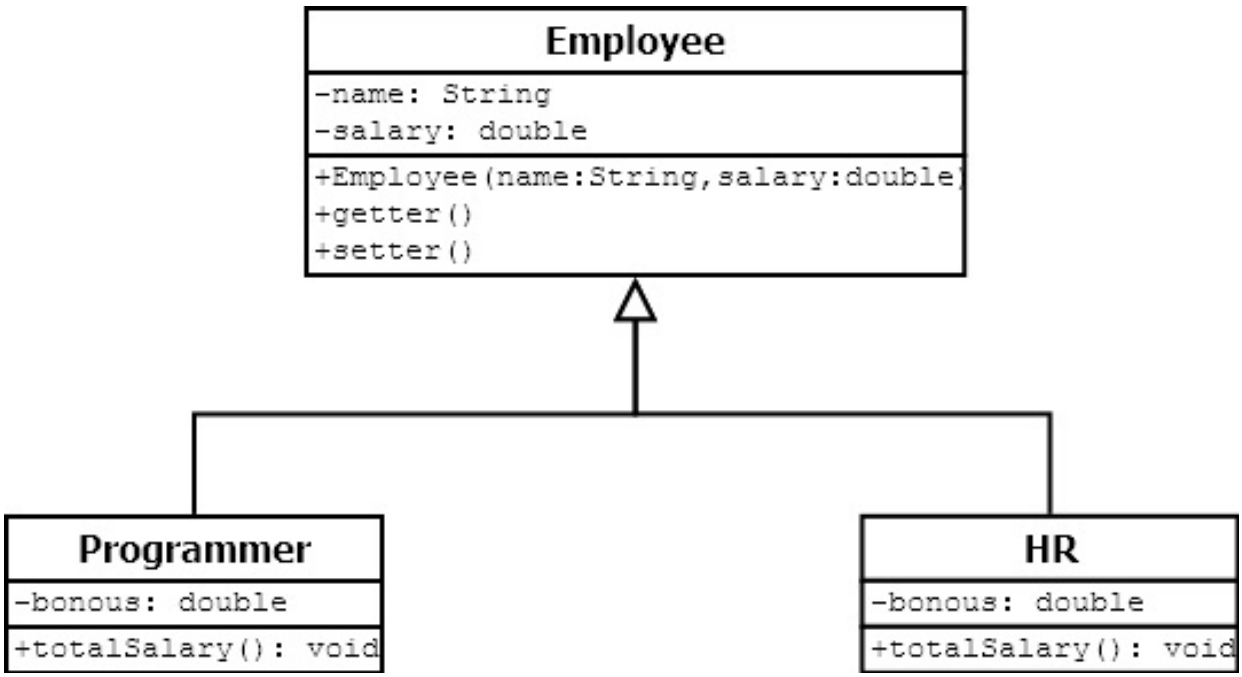
Runtime polymorphism using interface and method overriding



Compiletime polymorphism using method overloading



4. Inheritance

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
- create new classes that are built upon existing classes.
- Inheritance represents the IS-A relationship which is also known as a parent-child relationship.
- 


```
public class Employee {
```

```
    private String name;
    private double salary;
    public Employee(String name, double salary) {
        super();
        this.name = name;
        this.salary = salary;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

```
public class Programmer extends Employee{
    private double bonous;
    public Programmer(String name, double salary) {
        super(name, salary);
        this.bonous=bonous;
    }
```

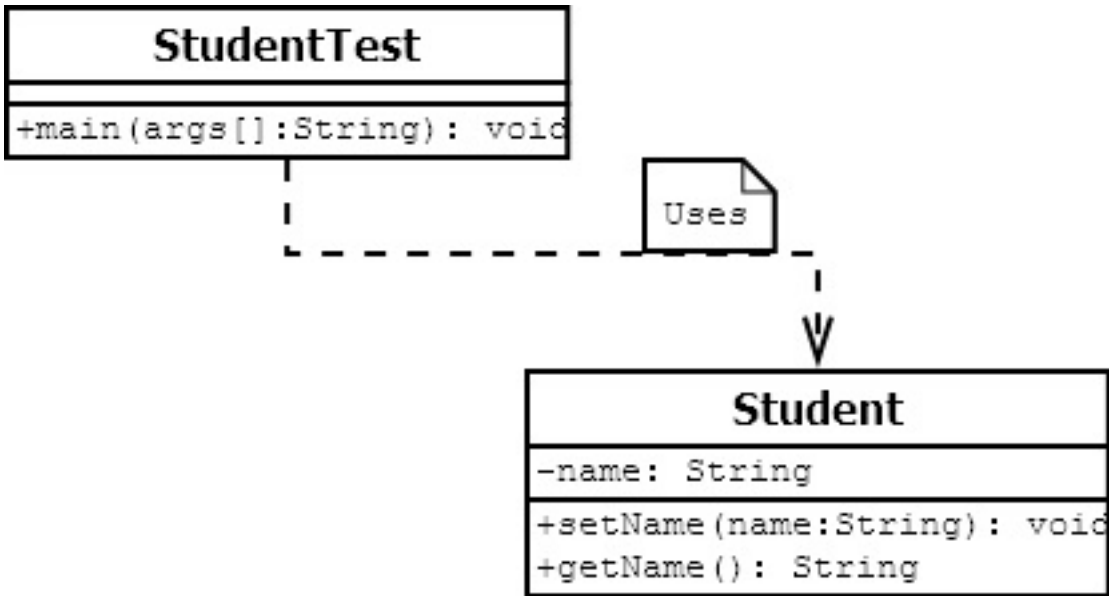
```
    public double getBonous() {
        return bonous;
    }
    public void setBonous(double bonous) {
        this.bonous = bonous;
    }
}
```

> *Watch the video tutorials*

> *Apart from these, there are some other concepts which are used in Object-Oriented design:*

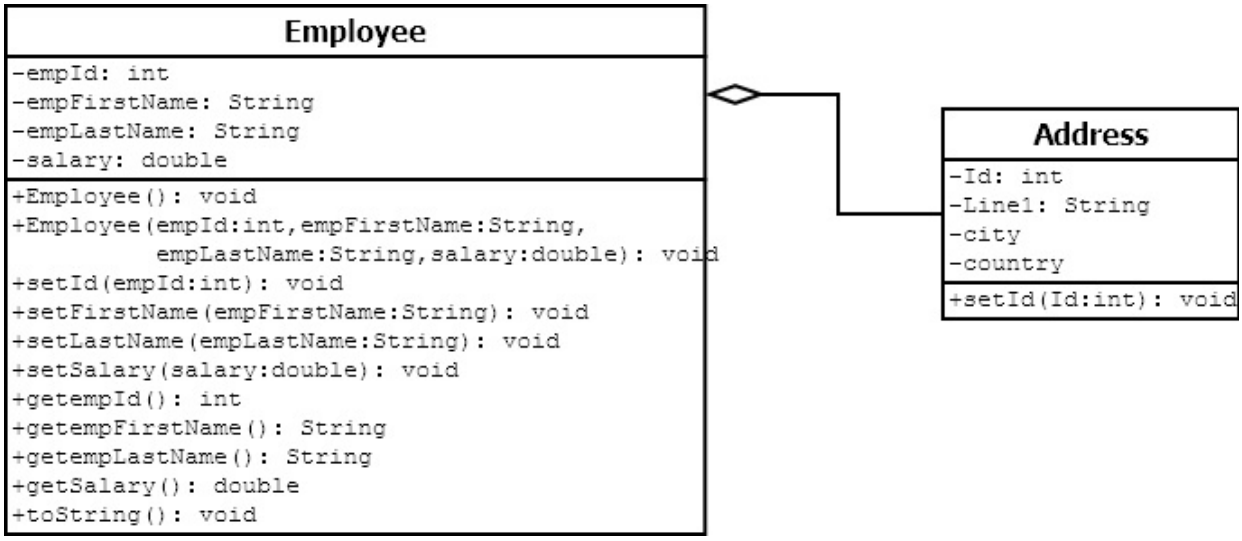
5. Association

- Association refers to the relationship between multiple objects.



5.1. Aggregation

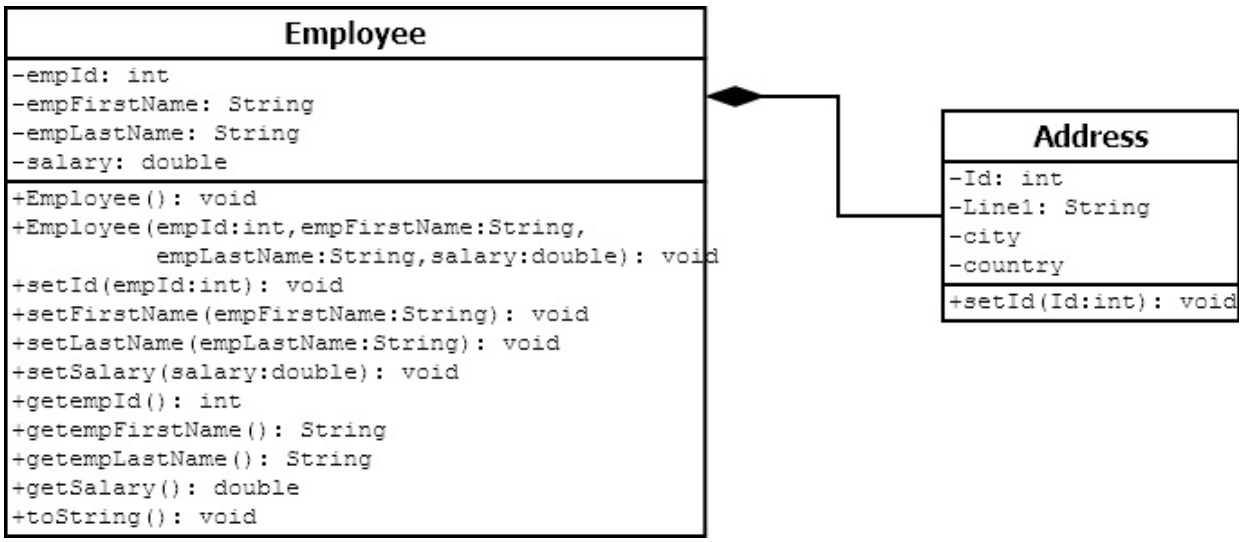
- Aggregation is a weak association.
- An association is said to be aggregation if both Objects can exist independently.



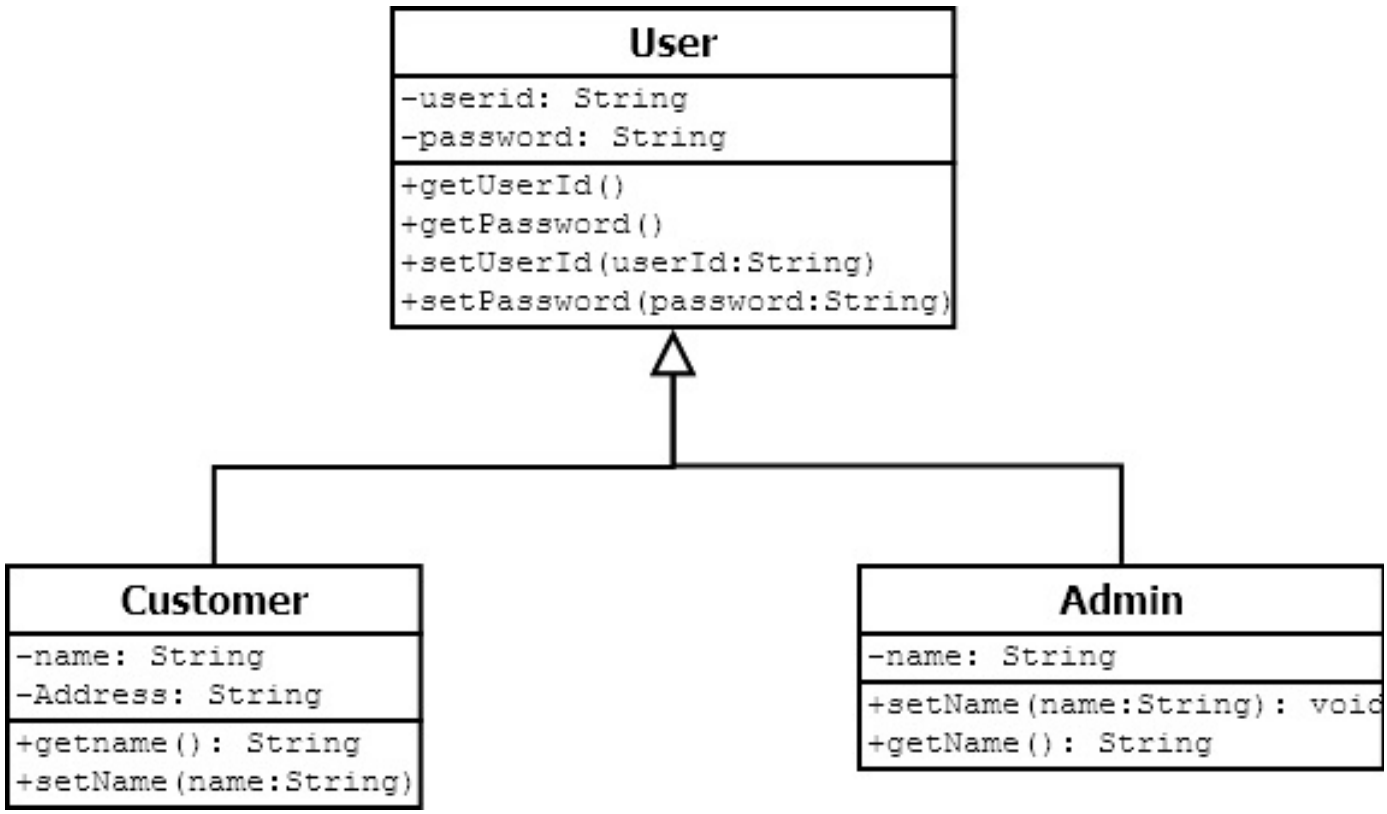
5.2. Composition

- The composition is the strong type of association.
- An association is said to composition if an Object owns another

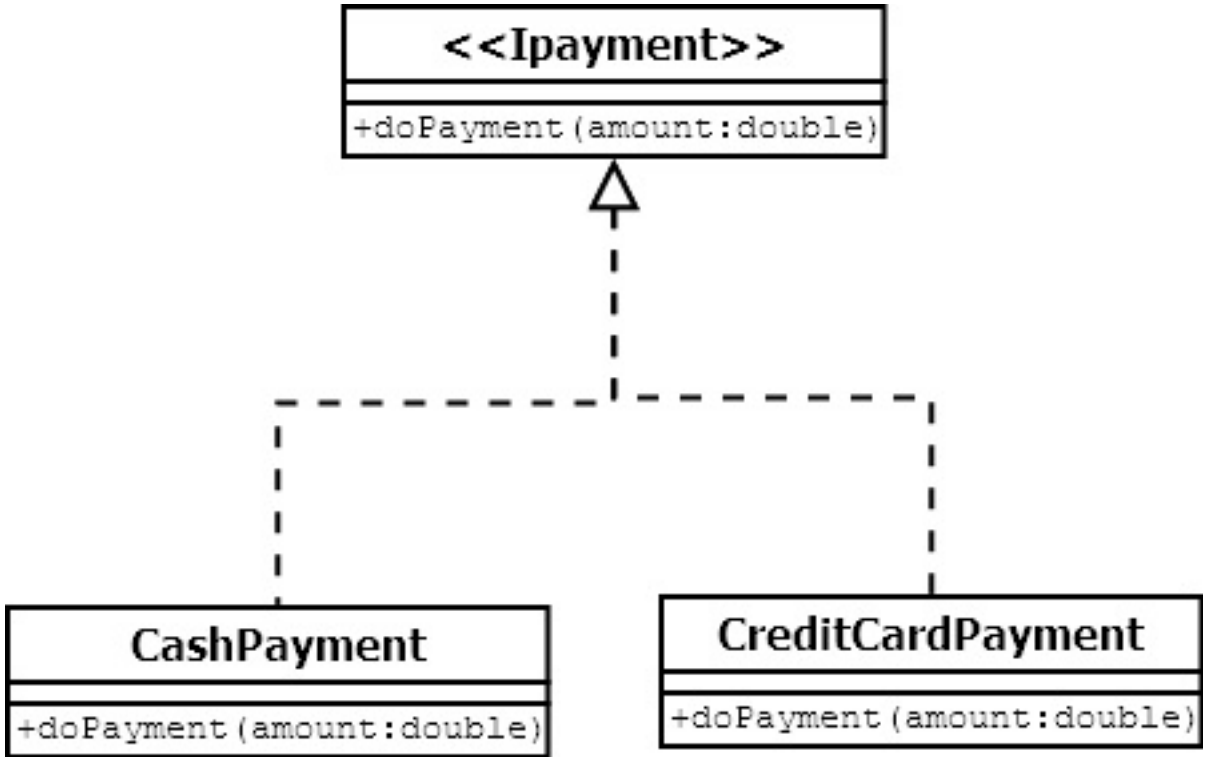
object and another object cannot exist without the owner object.



5.3. Generalization



5.4. Realization



6. Coupling

7. Cohesion