Lesson Proper for Week 13

Abstraction is the quality of dealing with ideas rather than events.

In Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Abstract Class

A class which contains the **abstract** keyword in its declaration is known as abstract class.

- Abstract classes may or may not contain *abstract methods*, i.e., methods without body (public void get();)
- But, if a class has at least one abstract method, then the class **must** be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Example

This section provides you an example of the abstract class. To create an abstract class, just use the **abstract** keyword before the class keyword, in the class declaration.

```java
/* File name : Employee.java */
public abstract class Employee {
  private String name;
  private String address;
  private int number;
  public Employee(String name, String address, int number) {
    System.out.println("Constructing an Employee");
    this.name = name;
    this.address = address;
    this.number = number;
  }
    public double computePay() {
   System.out.println("Inside Employee computePay");
   return 0.0;
  }
    public void mailCheck() {
    System.out.println("Mailing a check to " + this.name + " " + this.address);
  }
```

```java
  public String toString() {
    return name + " " + address + " " + number;
  }
  public String getName() {
    return name;
  }
   public String getAddress() {
    return address;
  }
   public void setAddress(String newAddress) {
    address = newAddress;
  }
   public int getNumber() {
    return number;
  }
}
```

You can observe that except abstract methods the Employee class is same as normal class in Java. The class is now abstract, but it still has three fields, seven methods, and one constructor.

Now you can try to instantiate the Employee class in the following way –

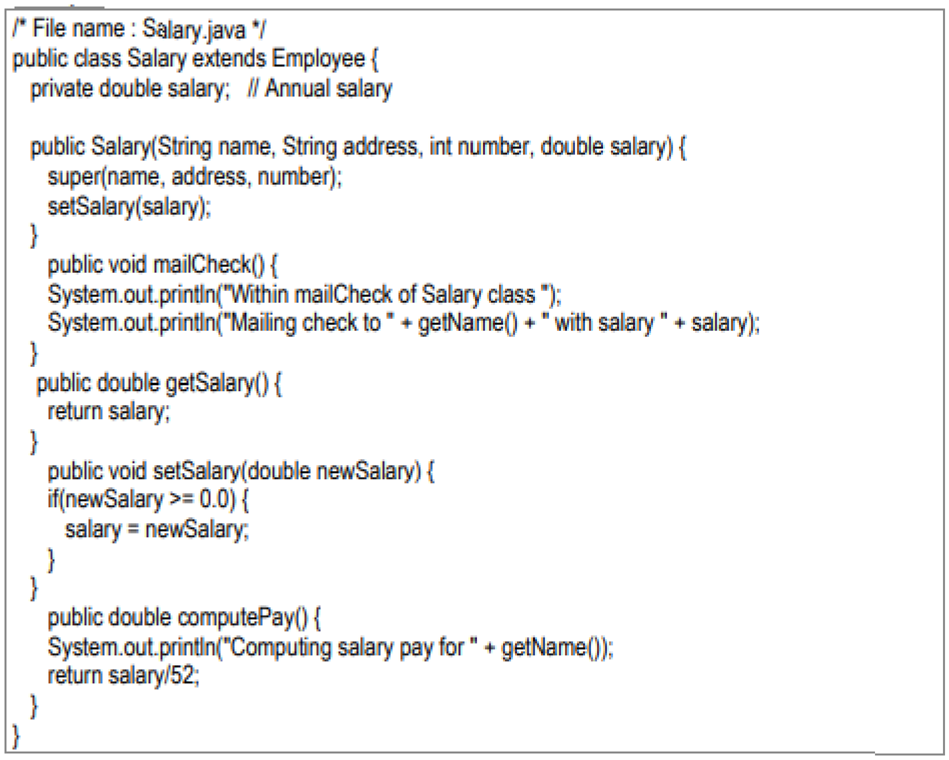
```java
/* File name : AbstractDemo.java */
public class AbstractDemo {

  public static void main(String [] args) {
    /* Following is not allowed and would raise error */
    Employee e = new Employee("George W.", "Houston, TX", 43);
    System.out.println("\n Call mailCheck using Employee reference--");
    e.mailCheck();
  }
}
```

When you compile the above class, it gives you the following error − Employee.java:46: Employee is abstract; cannot be instantiated Employee e = new Employee("George W.", "Houston, TX", 43); ^ 1 error

Inheriting the Abstract Class

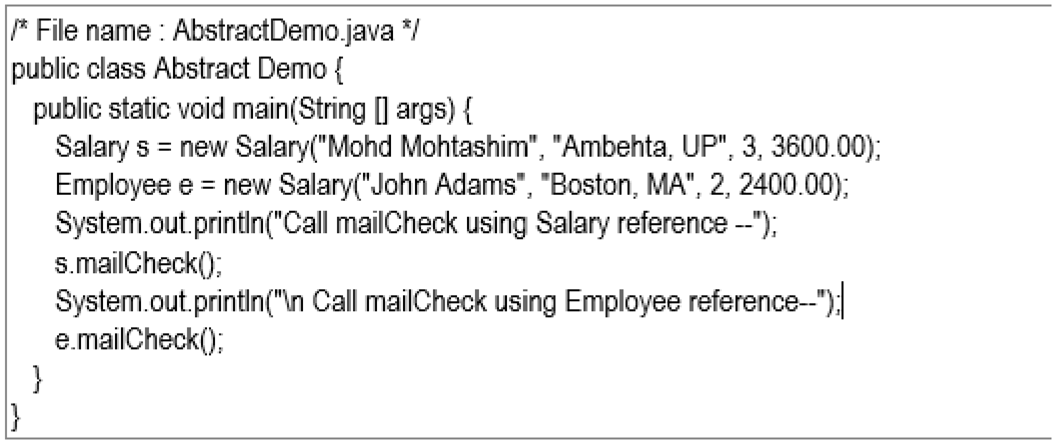
We can inherit the properties of Employee class just like concrete class in the following way −

Example



```
/* File name : Salary.java */
public class Salary extends Employee {
  private double salary;   // Annual salary

  public Salary(String name, String address, int number, double salary) {
    super(name, address, number);
    setSalary(salary);
  }
    public void mailCheck() {
    System.out.println("Within mailCheck of Salary class ");
    System.out.println("Mailing check to " + getName() + " with salary " + salary);
  }
   public double getSalary() {
    return salary;
  }
    public void setSalary(double newSalary) {
    if(newSalary >= 0.0) {
      salary = newSalary;
    }
  }
    public double computePay() {
    System.out.println("Computing salary pay for " + getName());
    return salary/52;
  }
}
```

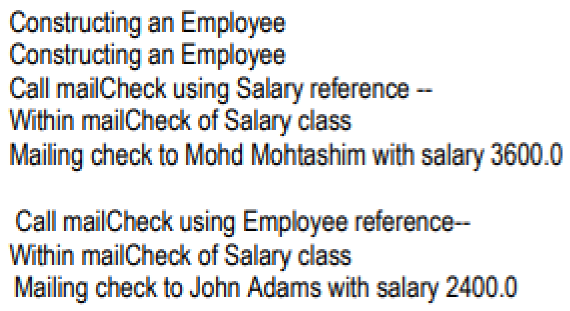
Here, you cannot instantiate the Employee class, but you can instantiate the Salary Class, and using this instance you can access all the three fields and seven methods of Employee class as shown below.



```
/* File name : AbstractDemo.java */
public class Abstract Demo {
  public static void main(String [] args) {
    Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00);
    Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00);
    System.out.println("Call mailCheck using Salary reference --");
    s.mailCheck();
    System.out.println("\n Call mailCheck using Employee reference--");
    e.mailCheck();
  }
}
```

This produces the following result −

Output



```
Constructing an Employee
Constructing an Employee
Call mailCheck using Salary reference --
Within mailCheck of Salary class
Mailing check to Mohd Mohtashim with salary 3600.0

 Call mailCheck using Employee reference--
Within mailCheck of Salary class
 Mailing check to John Adams with salary 2400.0
```

Abstract Methods

If you want a class to contain a particular method but you want the actual implementation of that method to determined by child classes, you can declare the method in the parent class as an abstract.

- **abstract** keyword is used to declare the method as abstract.
- You have to place the **abstract** keyword before the method name in the method declaration.
- An abstract method contains a method signature, but no method body.
- Instead of curly braces, an abstract method will have a semi colon (;) at the end.

Following is an example of the abstract method.

Example

```
public abstract class Employee {
   private String name;
   private String address;
   private int number;

   public abstract double computePay();
   // Remainder of class definition
}
```

Declaring a method as abstract has two consequences −

• The class containing it must be declared as abstract.

• Any class inheriting the current class must either override the abstract method or declare itself as abstract. Note − Eventually, a descendant class has to implement the abstract method; otherwise, you would have a hierarchy of abstract classes that cannot be instantiated. Suppose Salary class inherits the Employee class, then it should implement the compute Pay() method as shown below –

```
/* File name : Salary.java */
public class Salary extends Employee {
   private double salary;   // Annual salary

   public double computePay() {
      System.out.println("Computing salary pay for " + getName());
      return salary/52;
   }
   // Remainder of class definition
}
```

Advantages of Abstraction

Abstraction is a general concept which you can find in the real world as well as in OOP languages. Any objects in the real world, employees, or classes in your current software project, that hide internal details provide an abstraction.

These abstractions make it a lot easier to handle complexity by splitting them into smaller parts. In the best case, you can use them without understanding how they provide the functionality. And that not only helps you to split the complexity of your next software project into manageable parts, it also enables you every morning to brew a fresh cup of amazing coffee while you're still half asleep.

EXERCISE 1

A student is to create a simple Java program about person types. It includes student and employee. Create the person as an abstract and include abstract method name void activity();

____

____

____

____

EXERCISE 2

A student is to create a simple Java program about person types. It includes student and employee. Create the person as an abstract and include abstract method name void activity();, the student class and employee class must define the abstract class activity from the Person class with different method body.

____

____

____

____

◄ Preliminary Activity for Week 13

Jump to...

Analysis, Application, and Exploration for Week 13 ►

Navigation

Home

Dashboard

Site pages

My courses

121 - CC106

Participants

Grades

General

MODULE 1: WHAT IS APPLICATION DEVELOPMENT?

MODULE 2: WHAT ARE THE TECHNICAL SKILLS REQUIRED I...

COURSE OF ACTION: Whoever has maliciously uploaded these concerned materials are hereby given an ultimatum to take it down within 24-hours. Beyond the 24-hour grace period, our Legal Department shall initiate the proceedings in coordination with the National Bureau of Investigation for IP Address tracking, account owner identification, and filing of cases for prosecution.

2nd Semester Enrollment

visit www.bcp.edu.ph
Enrollment registration is now Ongoing
For 2nd Semester SY 2021 - 2022
We are accepting new students, returnees and transferees.
"Be trained to be the best, Be linked to success"
bcp-inquire@bcp.edu.ph
(8)442-8601 | (8)518-8050

Activities

- Assignments
- Forums
- Quizzes
- Resources