Lecture – 21

Abstract Classes and Methods

Abstraction

- Abstraction is the detail hiding or implementation hiding
- E.g. Driver need not to know internal detail of the break functionality.
- Abstraction can be achieved in two ways in JAVA
  - Abstract Classes
  - Interfaces

Abstract Class

- A class that has at least one abstract method is called an *abstract class*
  - An abstract class must have the modifier `abstract` included in its class heading:

```
public abstract class Employee
{
  private instanceVariables;
  . . .
  public abstract double earning();
  . . .
}
```

Abstract Classes

- Abstract classes can not be instantiated
- Abstract classes are declared with the keyword **abstract**

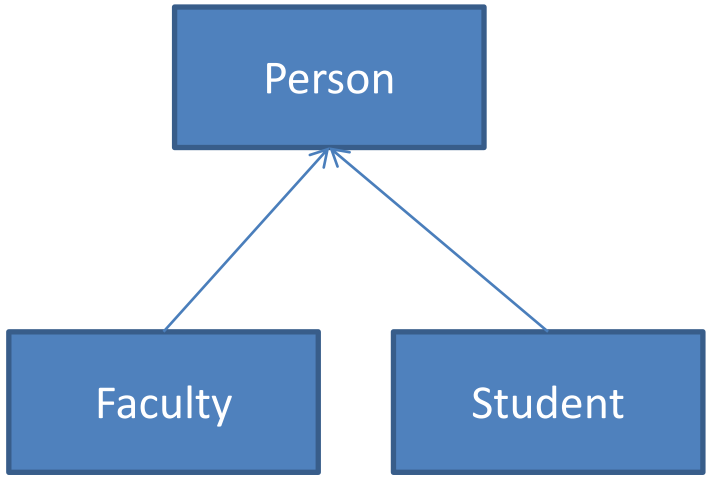
```
abstract class Person{
        private String name;
        private int age;

        public void setName(String n){ name = n;}
        public void setAge(int a){ age = a;}
}
```

```
public class Main{
        public static void main(String args[]){
                Person p = new Person();
        }
}
```

```
Person is abstract; cannot be instantiated
```

- *An abstract class can contain abstract methods, which are implemented in concrete subclasses.*

Abstract Classes



- Why we need abstract classes
  - To make class generalize
  - Don’t want a class whose object can be created
  - Person Class object is meaning less but Faculty and Student Objects are requirement of School Mgmt. System.
  - So Person Class is abstract class

Abstract Class

- Java abstract classes are use to declare common characteristics of subclasses
- It can only be used as superclass for other classes that extend the abstract class
- Like other classes, an abstract class can contain fields and methods
- Can not create object of abstract classes but we can create reference variable of abstract class

```
Person p = new Student();
```

Abstract Methods

- An abstract class can include methods that contain no implementation
- These are called abstract methods
- The abstract method declaration must then end with a semicolon rather than block

abstract void show();

- If a class has any abstract methods, whether declared or inherited, the entire class must be declared abstract

What is wrong with the code?

```
        class Person{
          abstract void show();
}
        class Student extends Person{
…
}
class Main{
          public static void main(String[] args){
                    Student s = new Student();
          }
}
```

Person class must be abstract because it contains abstract method

Since student class inherit method abstract void show(). Since it is still abstract so that's why the student class must be abstract.

Since Student class has become abstract. Then it can not be instantiated,

```
abstract class Person{
        abstract void show();
}
class Student extends Person{
        void show(){//some code}
}
class Main{
        public static void main(String[] args){
                Student s = new Student();
                s.show();
        }
}
```

Since Student class is overriding the show() method then Student class no more abstract.

In main we can make its object.

Why abstract methods?

- When a method don’t require implementation for its declaration
- Any class that contains an abstract method should be declared as an abstract class. Although the opposite might not be true i.e. it is not necessary that an abstract class should have an abstract method.
- Inheritance of an abstract class by a regular class requires the implementation of all the abstract methods in the parent class.

Points about abstract classes

- Abstract methods may or may not be present in the **Java abstract class**.
- The presence of at least one abstract method in a class makes the class an abstract class.
- An abstract class cannot have any objects and therefore cannot be directly instantiated.
- An abstract class can be used only if it is inherited from another class and implements the abstract methods.
- Proper implementations of the abstract methods are required while inheriting an abstract class.
- Parameterized constructors may be present in an abstract class. Also, an abstract class always contains a default constructor.