Module 1-13

Abstract Classes

Abstract Classes

Abstract Classes combine some of the features we've seen in interfaces with inheriting from a concrete class.

Abstract Classes

Abstract Classes combine some of the features we've seen in interfaces along with inheriting from a concrete class.

- Abstract methods can be extended by concrete classes.
- Abstract classes can have abstract methods
- Abstract classes can have concrete methods
- Abstract classes can have constructors
- Abstract classes, like Interfaces, cannot be instantiated

Abstract Classes : Declaration

We use the following pattern to declare abstract classes.

- The abstract class itself:

  public **abstract** class **<<Name of the Abstract Class>>** {...}

- The child class that inherits from the abstract class:

  public class **<<Name of Child Class>>** **extends** **<<Name of Abstract Class>>**

Abstract Classes Example

```
package te.mobility;

public abstract class Vehicle {

        private int numberOfWheels;
        private double tankCapacity;
        private double fuelLeft;

        public Vehicle(int numberOfWheels) {
                this.numberOfWheels = numberOfWheels;
        }
        public double getTankCapacity() {
                return tankCapacity;
        }

        public abstract Double calculateFuelPercentage();


        public double getFuelLeft() {
                return fuelLeft;
        }
}
```

We need to implement the constructor

We need to implement the abstract method

extends, not implement, is used.

```
package te.mobility;

public class Car extends Vehicle {

        public Car(int numberOfWheels) {
                super(numberOfWheels);
        }

        @Override
        public Double calculateFuelPercentage() {

                return super.getFuelLeft() / super.getTankCapacity() * 100;

        }

}
```

Also note how we are able to call concrete methods within the Vehicle abstract class

Abstract Classes: final keyword

Declaring methods as final prevent them from being overriden by a child class.

```
package te.mobility;

public abstract class Vehicle {
...

        public final void refuelCar() {
                this.fuelLeft = tankCapacity;
...
}
```

```
package te.mobility;

public class Car extends Vehicle {

        @Override
        public void refuelCar() {

        }

}
```

This override will cause an error, as the method is marked as final.

Multiple Inheritance

- Java does not allow multiple inheritance of concrete classes or abstract classes. The following is not allowed:

  public class Car **extends Vehicle, MotorVehicles** {...}

Where Vehicle and MotorVehicles are classes or abstract classes

- Java does allow for the implementation of multiple interfaces:

  public class Car **implements IVehicle, IMotorVehicle** {...}

Where IVehicle and IMotorVehicle are interfaces