

ABSTRACTION



IN THE NAME OF ALLAH

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

THE GRACIOUS, THE MERCIFUL.



GETS - GETTING EDUCATION WITH TECHNOLOGICAL SYSTEM

Instructor: Sir A.Rehman Ali Brohi

LECTURE: 10

Abstraction in java



- **OOP,s**
- **Class**
- **Objects & Methods**

- **Inheritance (IS-A)**
- **HAS-A relationship**
- **Polymorphism**
 1. Method Overloading
 2. Method Overriding

**Focus On
Code Reusability**

- **Abstraction**
- **Data Hiding**
- **Encapsulation**
- **Tightly Coupled Classes**

**Focus On
Security**

What is Abstraction?

Abstraction is hiding internal implementation & highlighting the setup services that we are offering.

Abstraction means to
show main services and
hide internal data

Real world Example the car, driver only need to see main dashboard not need to learn internal details like internal breaking system, gear system etc.



How to achieve abstraction in java

1. By Using **Abstract Class**. (0 % 100)
2. By Using **Interfaces**. (%100)

In this class we will
learn how to achieve
Abstraction using
Abstraction class

We will hide details

```
abstract Vehicle {  
    No_of_tyres;  
    abstract void start();  
}
```

```
Car {  
    No_of_tyres=4  
    start(){  
        System.out.println("Start with key");  
    }  
}
```

```
Scooter {  
    No_of_tyres=2  
    start(){  
        System.out.println("Start with kick");  
    }  
}
```

POINT TO NOTE:

```
Vehicle {  
    No_of_tyres;  
    abstract void start();  
}
```

1. A method without body (no implementation) is known as abstract method.

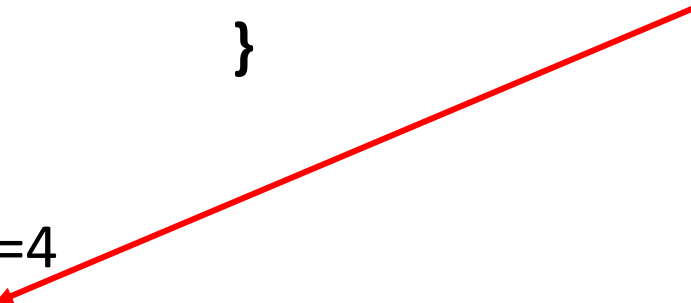
```
abstract Vehicle {  
    No_of_tyres;  
    abstract void start();  
}
```

2. A Method must always be declared in an abstract class, or we can say that if a class has an abstract method, it should be declared as well.


POINT TO NOTE:

```
abstract Vehicle {  
    No_of_tyres;  
    abstract void start();  
}
```

```
Car {  
    No_of_tyres=4  
    void start(){  
        System.out.println("Start with key");  
    }  
}
```



```
Scooter {  
    No_of_tyres=2  
    void start(){  
        System.out.println("Start with kick");  
    }  
}
```



3. If a regular class extends an abstract class, then the class must have to implement all the abstract methods of abstract parent class or it has to be declared abstract as well.

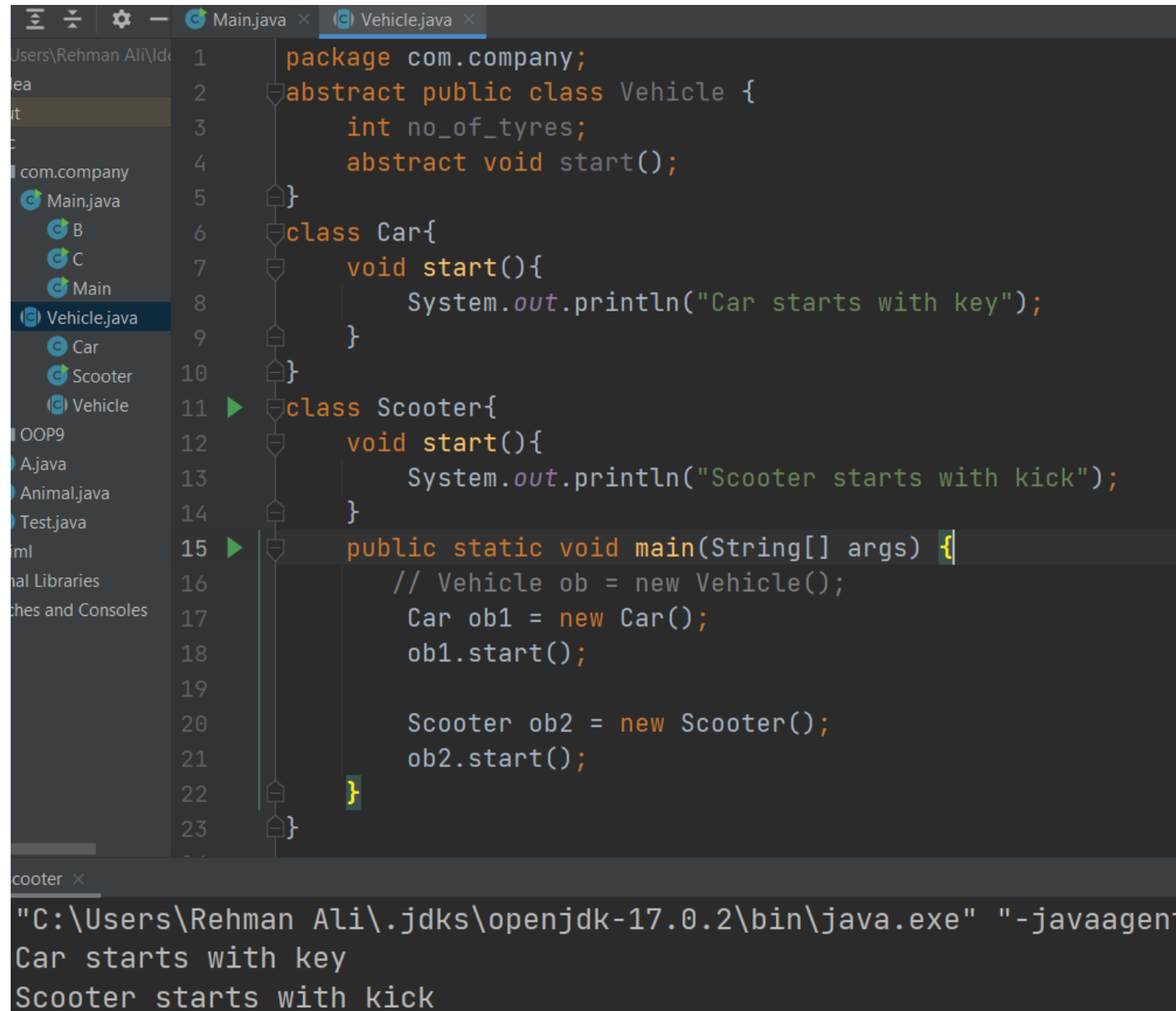
POINT TO NOTE:

Method Overriding

1. Same Method
2. Different Class
3. Same Arguments
 - No of Arg
 - Seg of Arg
 - Types of Arg
4. Inheritance (IS-A)

4. Abstract methods in an abstract class are meant to be **overridden** in derived concrete classes otherwise compile-time error will be thrown.
5. Abstract classes cannot be instantiated, means we **can't create an object** of **Abstract class**.

Program to create abstract class and abstract method with achieve overriding concept is known as abstract



```
1 package com.company;
2 abstract public class Vehicle {
3     int no_of_tyres;
4     abstract void start();
5 }
6 class Car{
7     void start(){
8         System.out.println("Car starts with key");
9     }
10 }
11 class Scooter{
12     void start(){
13         System.out.println("Scooter starts with kick");
14     }
15 }
16 public static void main(String[] args) {
17     // Vehicle ob = new Vehicle();
18     Car ob1 = new Car();
19     ob1.start();
20
21     Scooter ob2 = new Scooter();
22     ob2.start();
23 }
```

The screenshot shows an IDE with two tabs: Main.java and Vehicle.java. The Vehicle.java tab is active, displaying the code for an abstract class Vehicle and its subclasses Car and Scooter. The Car class overrides the start() method to print "Car starts with key", and the Scooter class overrides it to print "Scooter starts with kick". The main method in Main.java creates instances of Car and Scooter and calls their start() methods. The output at the bottom of the IDE shows the execution results: "Car starts with key" and "Scooter starts with kick".

What we have learn from this exercise

- What is Abstract Method.
- What is Abstract Class.
- Use of Abstraction
- How to achieve Abstraction by using abstraction class.