MD. Atif Rahman Rudro
IT-22002

Abstraction code

```
import java.io.PrintStream // needed when to
                           create or use a PrintStream object
                           explicitly.

// Interface: Defines engine behaviour that all
vehicles must implement for startEngine and
endEngine method.

interface Engine {

    PrintStream o = System.out;
    void startEngine();
    void endEngine();

}

//abstract class : Server as a base class for
    all vehicles.
```

```
abstract class Vehicle{
    String band;
    int year;
// Constructor: To set values of brand and year
since same variable is used for instantiation this
keyword is used as a reference.
    public Vehicle(String band, int year)
    {
        this.brand=brand;
        this.year=year;
    }
// Abstract method: Must be implemented by all
subclasses. Abstract methods don't print anything
    abstract void drive();
// Concrete method: Shared behaviour for all
vehicles.
public void displayInfo(){
Engine.o.println("Brand: "+brand+", Year: "+
                                    year);
```

```
    }
}

//subclass: Class Car is inherited from Vehicle
that implements Engine.
class Car extends Vehicle implements Engine
{
    //Constructor: Calls superclass constructor
    that sets brand and year;
    public Car(String brand, int year) {
        super(brand, year);
    }
    //Override: instructs the compiler that you
    intend to override the method in superclass.
    @Override
    //startEngine: implements startEngine method
    from Engine interface.
    public void startEngine() {
```

```
        o.println("Car Engine is starting...");
    }
    @Override
    // endEngine: implements endEngine from Engine
    interface.
    public void endEngine() {
        o.println("Car Engine is ending...");
    }
    @Override
    // Drive: Implements abstract drive method from
    vehicle class.
    public void drive() {
        o.println("Car is being driving...");
    }
}
// subclass: Bike is inherited from vehicle that
implements the Interface Engine
```

~~class Bike (String brand, int year)~~

class Bike extends Vehicle implements Engine

{

// Construction: Calls superclass constructor that sets brand and year.

public Bike(String brand, int year) {

super(brand, year);

}

@Override

// startEngine: implements startEngine method from Engine interface

public void startEngine() {

System.out.println("Bike Engine is starting...");

}

@Override

// endEngine: implements endEngine method from Engine interface;

public void endEngine() {

```
        o.println("Bike Engine is ending...");
    }

    @Override
    // Drive: Implements abstract drive method from
    Vehicle class.
    public void drive() {
        o.println("Bike is being driving...");
    }
}

//// Subclass: Boat is inherited from Vehicle that
implement the Interface Engine:
class Boat extends Vehicle implements Engine {
    // Constructor: Calls superclass constructor
    that sets brand and year.
    public Boat(String brand, int year) {
        super(brand, year);
    }
    @Override
```

```
//startEngine: implements startEngine method
 from Engine Interface

public void startEngine() {
    o.println("Boat Engine is starting...");
}

@Override
//endEngine: implements endEngine method
 from Engine interface.

public void endEngine() {
    o.println("Boat Engine is ending...");
}

@Override
//Drive: Implements abstract drive method from
  Vehicle class.

public void drive() {
    o.println("Boat is being sailing...");
  }
}
```

```
public class Abstraction {

    public static void main(String[] args) {

        Engine.o.println("I am MD. Atif Rahman Rudro-
                          My ID is IT:22002");

        Engine.o.println();
        //Create a Car object using a Vehicle reference
          (polymorphism)
        Vehicle car = new Car("X-Corolla", 2002);

        car.displayInfo(); // calls concrete method from
                              Vehicle
        ((Engine) car).startEngine(); // cast to Engine
                                         to access startEngine

        car.drive(); // calls overridden drive method
                        in Car

        ((Engine) car).endEngine(); // Cast to
                                       Engine to access stopEngine;

        Engine.o.println(); // Just a new line.
```

// Create a Bike object using a Vehicle reference (polymorphism)

Vehicle bike = new Bike("Hero-Honda", 2001);

bike.displayInfo(); // calls concrete method from Vehicle.

((Engine) bike).startEngine; // cast to Engine to access startEngine

bike.drive(); // calls overridden drive method in bike

((Engine) bike).endEngine(); // cast to Engine to access stopEngine

Engine.o.println(); // Just a line break

// Create a Boat object using a Vehicle reference (polymorphism)

Vehicle boat = new Boat("Osprey", 2022);

boat.displayInfo(); // calls concrete method from Vehicle

```
((Engine) boat).startEngine(); //cast to Engine
                    to access startEngine

boat.drive(); //calls overridden drive method in
                    Boat

((Engine) boat).endEngine(); //cast to Engine
                    to access endEngine;

    }
}
```

Output:

```
I am MD. Atif Rahman Rudro. My ID is IT-22002

Brand: X-Corolla, Year: 2002

Car Engine is starting...

Car is being driving...

Car Engine is ending...

Brand: Hero-Honda, Year: 2001

Bike Engine is starting...
```

Bike is being driving...

Bike Engine is ending...

Brand: Osprey, Year: 2022

Boat Engine is starting...

Boat is being sailing...

Boat Engine is ending...