Programming Test Results (With Test Cases)

Result Summary

Field	Value
Test ID	40156
Student ID	29195
Programs (with test cases)	1
Total Test Cases	2
Test Cases Passed	2
Fully Passed Programs	1
Partially Passed Programs	0
Failed Programs	0
Overall % (with test cases)	100.00%
Grade	Outstanding

Programs With Test Cases

#	Program Name	Total TC	Passed	Success Rate	Score /10	Submitted At	Attempts
1	Change the Driver scenario	2	2	100.0%	10	20/11/2025, 12:32:02	0

Program Details (With Test Cases)

Program 1: Change the Driver scenario

Languages: Java

Score (010): 10 / 10

Test Case Summary: Total: 2 Passed: 2
Failed: 0 Success: 100.0%

Attempts: 0

Submitted At: 20/11/2025, 12:32:02

Description:

Scenario Description :-

This is a scenario based program on Copy Constructor and Deep copy concept (another object) with HAS-A realtion.

It describes that if we create another object by using copy constructor and deep copy concept then Modification done in first object will not reflect another object that means another object will remain unchanged.

Coding Requirements :
==========
Create a BLC class called Driver with the following private non static fields:

-> name;
-> age;

Constructor :-

Implement a parameterized constructor to initialize all non static fields.
Implement getter and setter methods for all non static fields.

Create a BLC class Car with the following attributes:
-> private String brand;
-> private String model;
-> private int year;
-> private Driver driver;

Constructor :-

Implement a parameterized constructor to initialize all non static fields.

Implement a copy constructor (which accepts Car as a paraneter) for the Car and initializes the new Car object with the existing object properties. [Deep copy]

methods()

Implement getter and setter methods for all non static fields.

Implement a method named changeDriver(Driver newDriver) to update the driver of the car with public modifier and void as a return type.

Create two Car objects by using Deep copy concept.
Now, Change the Driver from first Car object by using changeDriver() method.
Verify drived changed in both the Car Objects OR only 1 object. [Deep Copy]

Step 1: Ask the user to choose a scenario
Press 1 Change the Driver of Car1 (Scenario 1)
Press 2 Do NOT change the Driver (Scenario 2)

Step 2: Take input from the user

User must enter:
Car brand
Car model
Car year
Driver name
Driver age
Create an ELC class UpdateDriverScenario with main method.

Constraints: -

Sample Input: Choice: 1 Car1 Brand: Toyota Car1 Model: Corolla Car1 Year: 2020 Driver1 Name: John Driver1 Age: 30 Car2: Deep Copy of Car1 New Driver Name: Mike New Driver Age: 40

Sample Output: === After Driver Update === Car 1: Toyota Corolla (2020), Driver: Mike (40) Car 2: Toyota Corolla (2020), Driver: John (30)

Explanation: For Executing the test cases take choice value as input first then take the inputs based on that for that you can use switch case or if else .

Solution Code

```
import java.util.*;
public class UpdateDriverScenario{
    public static void main(String [] args){

        Scanner sc = new Scanner(System.in);
        int choice = Integer.parseInt(sc.nextLine());
        String brand = sc.nextLine();
        String model = sc.nextLine();
        int year = Integer.parseInt(sc.nextLine());
```

```
        String name = sc.nextLine();
        int age = Integer.parseInt(sc.nextLine());

        switch(choice){
            case 1->{
                String  names = sc.nextLine();
        int age2 = Integer.parseInt(sc.nextLine());
                System.out.println("After Driver Update :");
               // Driver d = new Driver(name,age);
                Car c = new Car(brand,model,year,name,age);
                 Car d = new Car(brand,model,year,names,age2);
                 System.out.println(d.toString2());

                System.out.println(c.toString());
            }
            case 2->{
              System.out.println("No Driver Change :");
              Car c = new Car(brand,model,year,name,age);
              System.out.println(c.toString2());
              System.out.println(c.toString());
            }
        }

    }
}
class Driver {
   private String name;
    private int age;
    Driver(String name,int age){
        this.name=name;
        this.age=age;
    }
    public void setName(String name){
        this.name=name;
  }
  public String getName(){
    return this.name;
  }
  public void setAge(int age){
    this.age=age;
  }
```

```
  public int getAge(){
    return this.age;
  }
}

class Car{
    private String brand;
    private String model;
    private int year;
    private Driver dirver;

    Car(String brand,String model,int year,String name,int age){
        this.brand = brand;
        this.model = model;
        this.year=year;
        this.dirver=new Driver(name,age);
    }
    //Car(Car c){
      // Car d = new Car(String Brand,String model,int year);

   // }

   public void changeDriver(Driver newDriver){

   }
public String toString(){
    return "Car 2: "+this.brand+" "+this.model+" ("+this.year+")"+", Driver:
"+dirver.getName()+" ("+dirver.getAge()+")";
}
public String toString2(){
  return "Car 1: "+this.brand+" "+this.model+" ("+this.year+")"+", Driver:
"+dirver.getName()+" ("+dirver.getAge()+")";
}

}
```