

Enums

Objective

In this lab, we will look at creating enums in Java.

Overview

In this lab you will:

- Create an enum to represent tire pressures and look at how to override the values of the enum members.

Step by Step Instructions

Exercise 1: Create the enum

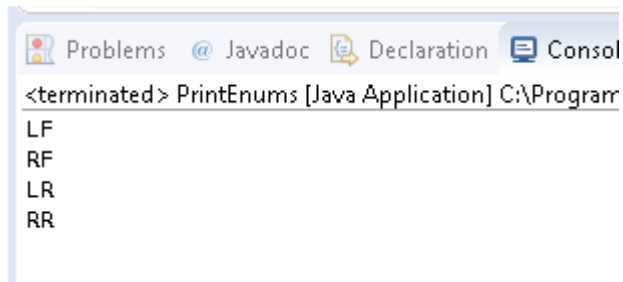
1. Create a new Java Project named **Enums**. In the **src** folder create two packages: **com.lq.enums** and **com.lq.app**.
2. Create a new **Enum** in the **com.lq.enums** package named **TirePressures** and implement it as follows:

```
package com.lq.enums;  
  
/**  
 * @author Student  
 *  
 */  
public enum TirePressures {  
    LF, RF, LR, RR;  
}
```

3. In the **com.lq.app** package, create a new driver class named **PrintEnums**. The code should look like this:

```
public class PrintEnums {  
    public static void main(String[] args) {  
        for (TirePressures tp: TirePressures.values()) {  
            System.out.println(tp);  
        }  
    }  
}
```

Run **PrintEnums**. You should see the names of the enums appear in the console.



```
<terminated> PrintEnums [Java Application] C:\Program
LF
RF
LR
RR
```

4. Tires need pressure, so add a constructor that passes the pressure for each tire. You will need to add an **int** named **pressure** to hold the pressure for each enum instance. Add a public getter for the pressure;

```
public enum TirePressures {
    LF(30), RF(30), LR(32), RR(32);
    int pressure;

    private TirePressures(int pressure) {
        this.pressure = pressure;
    }
}
```

5. Add a String attribute for **name** with a public String getName().
6. Add another parameter to the constructor for the name. Add values to each member to provide the parameter to the constructor call. Your code should be like this:

```

package com.lq.enums;

/**
 * @author Student
 */
public enum TirePressures {
    LF(30, "Left Front"),
    RF(30, "Right Front"),
    LR(32, "Left Rear"),
    RR(32, "Right Rear");

    private int pressure;
    private String name;

    private TirePressures( int pressure, String name) {
        this.pressure = pressure;
        this.name = name;
    }

    public int getPressure() {
        return pressure;
    }

    public String getName() {
        return name;
    }
}

```

7. In PrintEnums, change the main() method to print out the value of the tire pressure and name.

```

public class PrintEnums {
    public static void main(String[] args) {
        for (TirePressures tp: TirePressures.values()) {
            System.out.printf("%s is %d pounds%n",
                tp.getName(), tp.getPressure());
        }
    }
}

```

8. Create a new class in the **com.lq.app** folder name **EnumTester**. Make sure that the class has a **public void main(String[] args)** method.

```

package com.lq.app;
import com.lq.enums.TirePressures;

public class EnumTester {

    public static void main(String[] args){
        for(TirePressures t : TirePressures.values()) {
            System.out.println(t + " " + t.getPressure());
        }
    }
}

```

Though relatively rare, we might also want to override some of the pressures in the enum. Add an **overridePressure(int pressure)** method to the **TirePressures** enum.

```

public void overridePressure( int pressure) {
    this.pressure = pressure;
}

```

9. Now add code to the **PrintEnums** class to find the “RR” member of the **TirePressures** enum using the enum **valueOf(String name)** method. Once the member has been returned, use the **overridePressure** method to change the tire pressure for that member and then loop through the enum to display the pressures for all the members. Your code should resemble the following:

```

public static void main(String[] args){

    TirePressures tp = TirePressures.valueOf("RR");
    tp.overridePressure(22);

    for(TirePressures t : TirePressures.values()) {
        System.out.println(t + " " + t.getPressure());
    }
}

```

Test your code and you should see that the pressure for the right rear has now been changed from the original pressure of 32 to a new pressure of 22.

10. The members of an enum can be reference statically. You could add a static import for all the members of the **TirePressures** enum in the **PrintEnums** class.

```
import static com.lq.enums.TirePressures.*;
```

and you could then use it in your method:

```
TirePressures rr = valueOf("RR");
```

But does that look like good, maintainable, code to you? And what if you were using more than one Enum? Static imports can be good and bad.

11. Challenge Exercise: Consider that your company only did business in three states? How would you limit addresses to those three states? Add full state name and state capitol to your answer. Code your answer with a driver to prove.