Homework – Java 005

Enumerations and Generics

Root folder: java_training/assignments/java_005

Assignments

- Traffic Light Simulator
- Generic Array Swap

Traffic Light Simulator

folder: trafficlight

Quick Description

Traffic light using enumeration and multithreading and synchronization

Rationale

- Enumerations are useful when your program needs a set of constants, but the actual values of the constants are arbitrary, as long as all differ.
- One common instance involves handling the states in which some device can exist.
- Imagine that you are writing a program that controls a traffic light. Your traffic light code must automatically cycle through the light's three states: green, yellow, and red.
- It also must enable other code to know the current color of the light and let the color of the light be set to a known initial value.
- This means that the three states must be represented in some way.
- Although it would be possible to represent these three states by integer values (for example, the values 1, 2, and 3) or by strings (such as "red", "green", and "yellow"), an enumeration offers a much better approach.
- Using an enumeration results in code that is more efficient than if strings represented the states and more structured than if integers represented the states.

Assignment

Write a program that uses an enumeration to hold data.

Use a loop that will break and terminate the app when pressing Enter.

While the loop is executing, light state will keep cycling for a pre-determine amount of seconds, depending on the light enumeration object.

You will use a thread to control the lights

Upon Starting your program

Display Text: "Traffic Light Simulator is a Go!"

Cycle between these 3 states (display as message)

- Light is Green - Go!
- Light is Yellow - Beware!
- Light is Read - Stop!

Pressing Enter, the program will exit, but you will output this ending message

"Traffic Light Simulator is Done!"

The amount of cycle per light to use

- Green = 5 seconds
- Yellow = 2 seconds
- Red = 3 seconds

Extra work (Java 005) - Optional

Create a class which can use the **enumeration** as a means of creating calculations.

Create an enumeration of US Dollar Coins

- penny
- nickel
- dime
- quarter
- half dollar
- dollar

So, you would have a class to handle the overall "Money Object"

```
Money.setPenny()
Money.setNickel()
...
```

Note: internally, this class handles the enum.

Another class to handle the **enumeration**

For example

CountMoney cm = new CountMoney(); <-- this class knows how to deal with output and potential gotchas and errors

And your Money object could be passed into your cm.setMoney(money)

So, set you want something like this:

```
money = Money.setPenny(2);
money = Money.setNickel(2);
money = Money.setDime(2);
money = Money.setDollar(2)
```

```
cm.setMoney(money)
```

The return would be:

```
cm.returnRaw() > 2.32
cm.returnPretty() > "2 dollars and 32 cents"
cm.returnCurrenty() > $.2.32
```

cm.setMoney() <-- no parameters returns

cm.returnRaw() > 0
cm.returnPretty() > "no money has been set"
cm.returnCurrency > "no money has been set"

Use custom exception handlers to ensure you have the correct 'error' handling

Encapsulate in the class the ability to ensure we can short form the **enumeration** (? static import)

Generic Array Swap

folder: genarrayswap

Assignment

Write a generic method to exchange the positions of two different elements in an array. It will swap elements of same data type. Use “generics” for implementation.