Building Reusable Components

INT101 Programming Fundamentals

Basic Programming Constructs

Syntax: Class

```
public class YourClassName {
    private static final Dt0 CONSTANT = …; // static final variables are constants
    private static Dt1 classVariable;       // static variables are class variables
    private Dt2 instanceVariable;       // non-static variables are instance variables

    // method that has the same name as the class name is called Constructor
    public YourClassName() {
        …
    }

    // method that is static is called class method
    public static Dt3 classMethod(Dt4 argument0) {
        Dt5 localVariable;
        …
    }

    // method that is not static is called instance method
    public Dt6 instanceMethod(Dt7 argument1) {
        Dt8 localVariable;
    }

}
```

Switch class 

- Write a class that behaves like an on/off switch
  - Each switch has an on/off state.
  - Each switch knows its current state (on or off).
  - Each switch can turn on, turn off, or toggle (reverse its state).
  - Each switch has an unchangeable name.

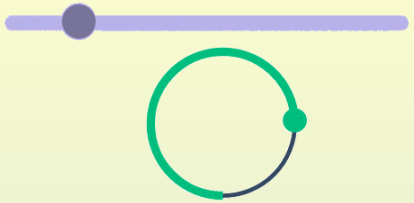
public class Switch 

- **`Switch(String name)`**
  - Constructor: set the unchangeable **`name`** of the **`Switch`** and initialize the state of the **`Switch`** to on.
- **`isOn() : boolean`**
  - Check if the **`Switch`** is on or not. Return **`true`** if it is on; otherwise return **`false`**.
- **`turnOn() : void`**
  - Turn the **`Switch`** on.
- **`turnOff() : void`**
  - Turn the **`Switch`** off.
- **`toggle() : void`**
  - Turn the **`Switch`** on if it is off, or turn the **`Switch`** off if it is on.
- **`toString() : String`**
  - Return the **`name`** of the **`Switch`**; and **`"(on)"`** if the **`Switch`** is on, or **`"(off)"`** if the **`Switch`** is not on.

Method Chaining

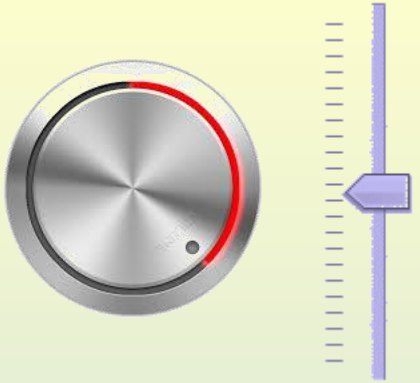
- Change the following methods to return **`this`**, instead of **`void`**.
  - **`turnOn()`**
  - **`turnOff()`**
  - **`toggle()`**
- So that it can be used in this fashion:
  - switch.on().toggle().off()

Slider class



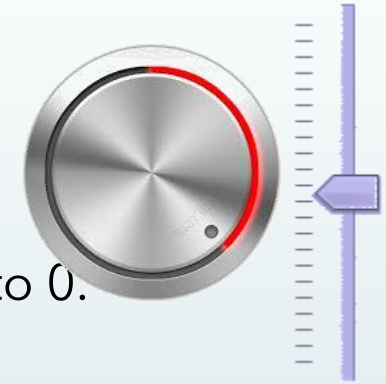
Bounded Slider

Circular Slider

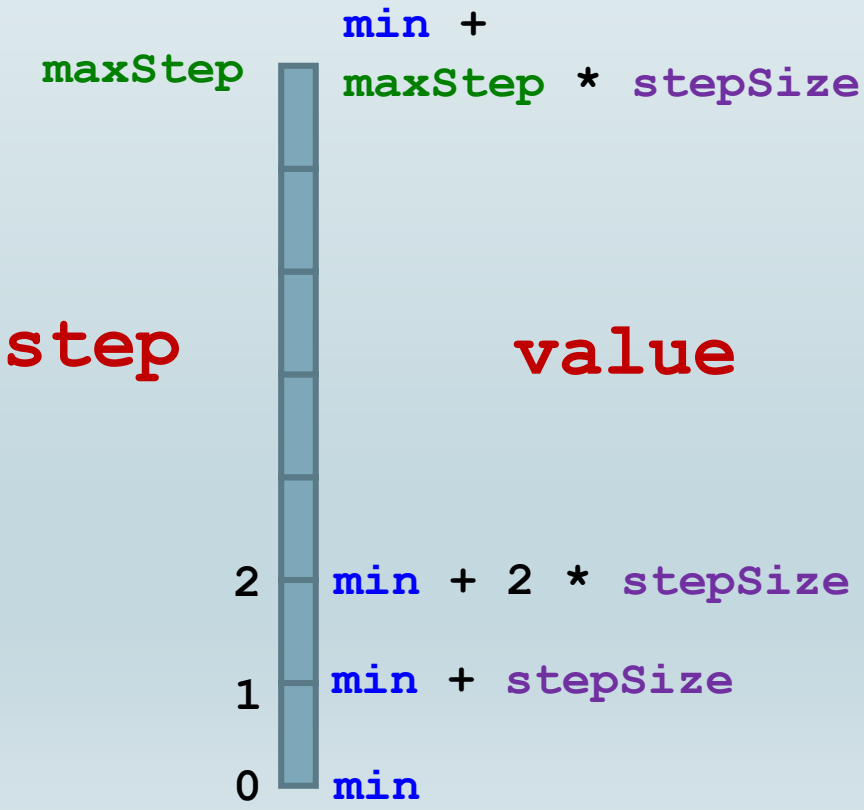


- Write a class that behaves like a slider
  - Each slider has a min value and a max value that it can be.
  - Each slider knows its current state (value).
  - Each slider can move the value up and move the value down.
  - A bounded slider cannot move the value beyond the min and max boundary.
  - For any circular slider,
    - if it moves up beyond the max value, its value will become the min value and
    - if it moves down under the min value, its value will become the max value
  - Each slider has an unchangeable name.

public class Slider



- **`Slider(String name, double min, double stepSize, int maxStep, boolean circular)`**
  - **Constructor**: set the unchangeable **`name`** of the **`Slider`** and set **the current step** to 0.
  - **`min`** = the minimum value that Slider;
    `stepSize` = the amount that the value will change for each modification;
    `maxStep` = the highest step that the value of the slider can be;
    `circular` = true if it is a circular slider; false if it is not.
- **`up() : void`**
  - Change the value -> one step up.
  - If it reaches the max value, it cannot go further;
    but if it is a circular slider, it will go down to the min value.
- **`down() : void`**
  - Change the value -> one step down.
  - If it reaches the min value, it cannot go further;
    but if it is a circular slider, it will go up to the max value.
- **`getValue() : double`**
  - **`Return min + current_step * stepSize`**
- **`toString() : String`**
  - Return the **`name`** of the **`Slider`**; and **`"(the value)"`**.



Radio class

- Write a class that behaves like a Radio
  - Each Radio has an on/off Switch
  - Each Radio has a volume Slider which is a non-circular one.
  - Each Radio has a station Slider which is a circular one.

  - If the Radio is off, it can neither change volume nor station.