Setting up event-driven UI

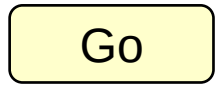
COMP 102

Victoria University of Wellington

Setting up event-driven input

- Setting up the GUI:
  - To add a button to the UI:
    - specify name of button and method to call (*object* ::*method* or *class* ::*method*)
      (must be a method with no parameters)

    eg: UI.addButton("Go", this::startGame);
        UI.addButton("End", UI::quit);




```
public void startGame(){…..
```

  - To add a textfield to the UI:
    - Specify name of textfield and method to call
      (must be a method with one String parameter)

    eg UI.addTextField("name", this::setName);

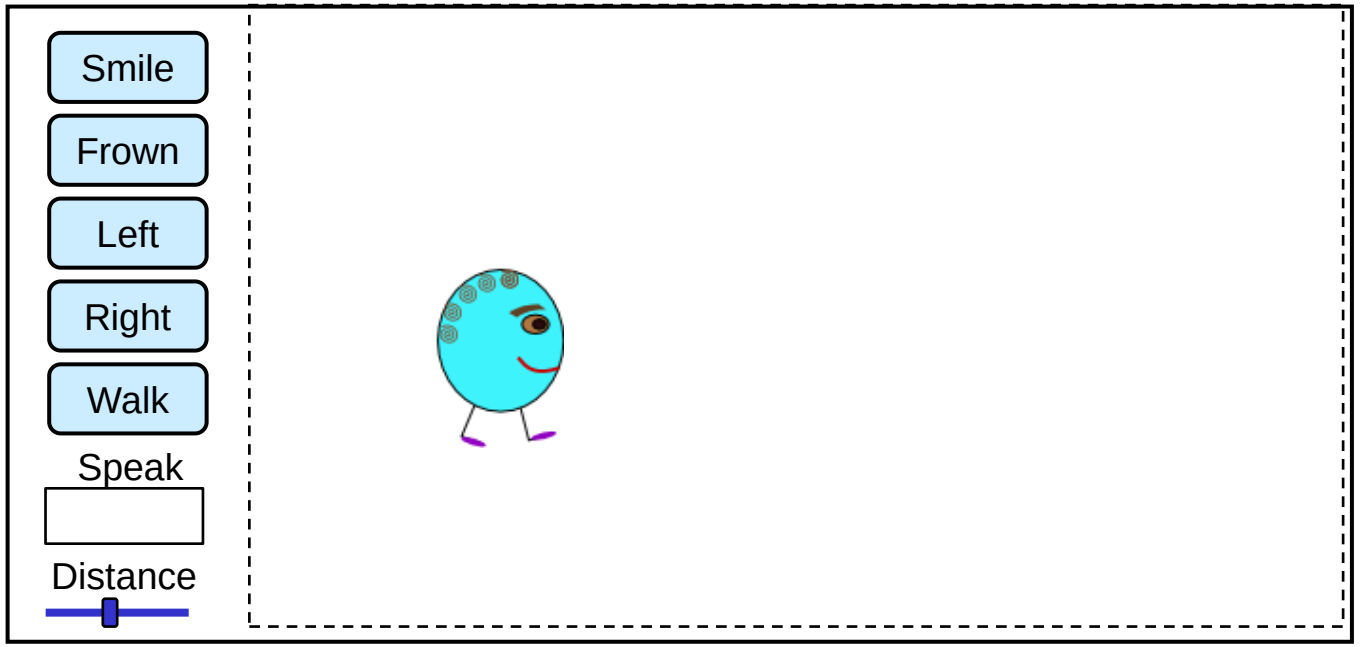

```
public void setName(String n){…..
```

  - To add a slider to the UI:
    - Specify name of slider, min, max, initial values, and method to call
      (must be a method with one double parameter)

    eg UI.addSlider("speed", 10, 50, 20, this::setSpeed););


```
public void setSpeed(double v){…..
```

PuppetMaster

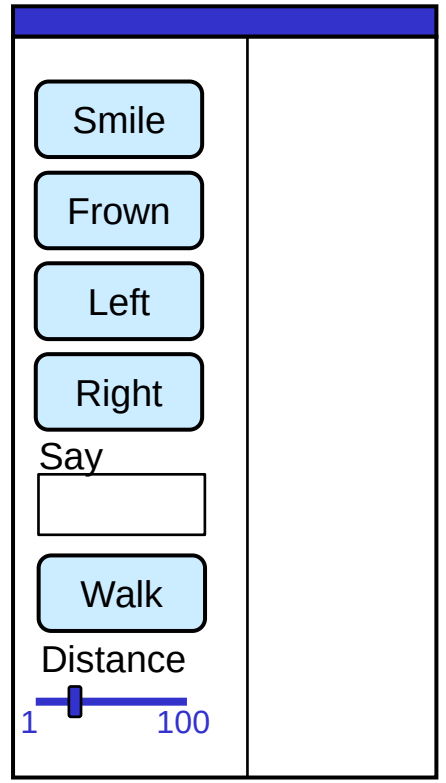


PuppetMaster: setting up Buttons etc

```
public class PuppetMaster   …   {
   // fields

   /** set up the GUI */
   public void setupGUI (){
      UI.addButton( "Smile", this::doSmile);
      UI.addButton( "Frown", this::doFrown);
      UI.addButton( "Left", this::doLeft);
      UI.addButton( "Right", this::doRight);
      UI.addTextField( "Say", this::doSpeak);
      UI.addButton( "Walk", this::doWalk);
      UI.addSlider( "Distance", 1, 100, 20, this::setDist);
            …
   }
   // methods to respond

   public static void main (String[ ] args){
      new PuppetMaster().setupGUI();
}
```



Responding to buttons and textFields

```
public class PuppetMaster {
    public void doSmile(){
        // tell the CartoonCharacter to smile
    }
    public void doFrown(){
        // tell the CartoonCharacter to frown
    }
    public void doSpeak(String words){
        // tell the CartoonCharacter to say the words
    }
    public void setDist(double value){
        // remember the value
    }
    public void setupGUI(){
        UI.addButton("Smile", this::doSmile);
        UI.addButton("Frown", this::doFrown);  …..
        UI.addTextField("Say", this::doSpeak);
        UI.addSlider( "Distance", 1, 100, 20, this::setDist);
    }
```

Methods called by buttons must have no parameters

Methods called by a textField must have one String parameter

Methods called by a slider must have one double parameter

Where will the CartoonCharacter be stored?

Event driven input and fields

- Each event will make a new method call.
- ⇒ can't remember anything between events in local variables in the methods.
-
- Typically, need fields in the object to remember information between events.
  - eg: PuppetMaster has to remember the CartoonCharacter object in a field

PuppetMaster: Design

Structure of the PuppetMaster class:

```
public class PuppetMaster  {

    // fields to store values between events/method calls
    private ….

    // set up GUI
    public void setupGUI() {
        // set up the buttons, slider, textField, to call methods on the object
    }

    // methods to respond to the buttons,  slider, textField
    public  void …


    public static void main (String[] args){
        // make a PuppetMaster object and call setupGUI
    }
```

PuppetMaster: Using Fields

Actions on the CartoonCharacter happen in response to different events

⇒ will be in different method calls

⇒ need to store character in a field, not a local variable.

```
public class PuppetMaster{
    // fields
    private CartoonCharacter cc = new CartoonCharacter(200, 100, "blueguy");

    public void doSmile(){
        this.cc.smile();
    }
    public void doFrown(){
        this.cc.frown();
    }
    public void setupGUI(){
        UI.addButton("Smile", this::doSmile);
        UI.addButton("Frown", this::doFrown);
          ⋮
    }
```

PuppetMaster: TextFields (boxes)

```
public  class  PuppetMaster{
    private CartoonCharacter cc = new CartoonCharacter(200, 100, "blueguy");

    public void doSmile(){
        this.cc.smile();
    }
            :
    public void doSpeak(String words){
        this.cc.speak(words);
    }

    public void setupGUI(){
        UI.addButton("Smile", this::doSmile);
        UI.addButton("Frown", this::doFrown);

        UI.addTextField(“Say", this::doSpeak);
    }
```

PuppetMaster: Sliders

```
public  class  PuppetMaster {
    private CartoonCharacter cc = new CartoonCharacter(200, 100, "blueguy");
    private double walkDist = 20 ;

    public void doWalk() {
        this.cc.walk(this.walkDist);
    }
    public void setDist(double value){
        this.walkDist = value;
    }
    public void setupGUI(){
        UI.addButton("Smile", this::doSmile);
        UI.addButton("Frown", this::doFrown);

              :
        UI.addButton("Walk", this::doWalk);
        UI.addSlider( "Distance", 1, 100, 20, this::setDist);
    }
              .
```

Typical design:
field to store value
from one event,
for use by another event

A method called by
a slider must have
one double parameter

PuppetMaster: Using Fields

Listeners in the buttons etc don't *have* to call methods on this or UI:

```
public  class  PuppetMaster{
    // fields
    private CartoonCharacter cc = new CartoonCharacter(200, 100, "blue");
    // constructor
    public void setupGUI(){
        UI.addButton("Smile",  this::doSmile);
        UI.addButton("Frown", this::doFrown);
            :
    }
    public void doSmile(){
        this.cc.smile();
    }
    public void doFrown(){
        this.cc.frown();
    }
```

PuppetMaster: Using Fields

Listeners in the buttons etc don't *have* to call methods on this or UI:

```
public  class  PuppetMaster{
    // fields
    private CartoonCharacter cc = new CartoonCharacter(200, 100, "blue");
    // constructor
    public void setupGUI(){
        UI.addButton("Smile",  cc::smile);
        UI.addButton("Frown", cc::frown);
            :
    }
    public void doSmile(){
        this.cc.smile();
    }
    public void doFrown(){
        this.cc.frown();
    }
```