Structure of a Java program

COMP 102

Victoria University of Wellington

Structure of a program

- Humans are phenomenal at understanding incomplete messages and find patterns in ill-structured material
  - Computers are generally *not* good at it
- Consequently, programs are written following strict rules.
  - For most languages (e.g. Java) that includes rules about the structure of the program

Structure of a Java program

Program Structure:

- Import
  - list the "libraries" the program will use
  - In COMP102 we always use ecs100 and commonly java.awt.Color and java.util.
- Class
  - Top-level component of a program
  - Describes a class of objects
    - Specifies the set of actions this kind of object can perform
    - Will also specify information the objects can store
  - When a Java program runs, there will always be one object, but there commonly are *many*
    - The class is a “recipe” to make one object
    - A program will typically have many classes that define objects that interact
- Methods
  - Main elements of a class
  - Each method describes an action that the objects of this class can perform

```java
import ecs100.*;
/** Program for converting between temperature scales */
public class TemperatureCalculator
{
    /** Print conversion formula */
    public void printFormula ( ) {
        UI.println("Celsius = (Fahrenheit - 32) *5/9");
    }

    /** Ask for Fahrenheit and convert to Celsius */
    public void doFahrenheitToCelsius(){
        double fahrenheit = UI.askDouble("Farenheit:");
        double celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
        UI.println(fahrenheit +  " F -> " + celsius + " C");
    }

    /** Setup the buttons */
    public void setupGUI(){
        UI.addButton("Formula", this::printFormula);
        UI.addButton("Convert", this::doFahrenheitToCelsius);
    }
}
```

Actions in a program

- Methods defines actions. To perform the action a method is “called”
- Method calls: *object* . *method* (*arguments*)
  - telling an object to do one of its methods, passing the necessary information as arguments:

```
UI.println("Celsius = (Fahrenheit - 32) *5/9");
UI.addButton("Formula", this::printFormula);
```

  - What are the possible objects? what are the possible methods.
    - **UI** object has methods for
      - Printing, asking, drawing, buttons, ….
    - **this** object – the one we are defining – has the methods being defined in the class
- Assignment statements *place* = *value*
  - Action: putting a value in a place (variable), so we can access it later

```
double fahren= UI.askDouble(“Fahrenheit:");
double celsius = (fahren – 32.0) * 5.0 / 9.0;
```

Elements of the program

- Comments *vs* Code
- Keywords / Identifiers / Strings / Types / numbers / operators and punctuation
  - **Keywords** : words with special meaning in the Java Language
    eg: **public**, **class**, **if**, **while**, …
    mostly to do with the structure of the program
  - **Identifiers** : other words, used to refer to things in the program.
    mostly made up by the programmer,
    some are predefined.
  - Strings : bits of text that the program will manipulate.
    always surrounded by " and "
  - Types : names for kinds of values.
  - **numbers**
  - **operators** and **punctuation** : + - * / = % . ; , ( ) { } [ ] ' "
    all have precise meanings and rules for use

```
import ecs100.*;
/** Program for converting between temperature scales */
public class TemperatureCalculator
{
    /** Print conversion formula */
    public void printFormula ( ) {
        UI.println("Celsius = (Fahrenheit - 32) *5/9");
    }

    /** Ask for Fahrenheit and convert to Celsius */
    public void doFahrenheitToCelsius(){
        double fahrenheit = UI.askDouble("Farenheit:");
        double celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
        UI.println(fahrenheit +  " F -> " + celsius + " C");
    }

    /** Setup the buttons */
    public void setupGUI(){
        UI.addButton("Formula", this::printFormula);
        UI.addButton("Convert", this::doFahrenheitToCelsius);
    }
}
```