Introduction to Programming (COSC 1519)

Lecture 5

Semester 1 - 2018

Lecturer: daryl.dsouza@rmit.edu.au

Author: gayan.wijesinghe@rmit.edu.au

Reflection

Explain in your own words…

Week 1: Intro to data types

Week 2: Variables and sequence

Week 3: if-statements (checking) and while-loops (repeating)

Week 4: Arrays and intro to data structures

And this week...

Week 5: Object variables and Methods (OO Part A)

Which one of these is showing a race?

Programming paradigms: Different ways to program

The Need for OO Design

- So far, we have put all of our data (e.g. variables, arrays, etc.) and instructions (declarations, if-statements, loops, etc.) in one method: `public static void main`

  This is commonly referred to as ***monolithic*** design. It's ok for small programs.

- But, bigger programs → Harder to get your head around the whole program at once (why?), some parts might be repeated (e.g. ?), different programmers will likely be working together, etc.

- Object oriented programming (OOP or OO) is solving problems by breaking them down in to classes and their objects.
  - A class is like a data type that we create. E.g. a multiplayer game can have one class named Player (in Eclipse it'll show as Player.java).
  - A class specifies how to create an actual instance of that class including what data (a.k.a. "object properties"/"instance variables"/"object variables"/"member variables") it has and what actions the object can perform (methods).
  - An object is a specific instance of the Player class. E.g. if there are 20 players in the game, we will have 20 Player objects and they can be named player1, player2, … player20.
    - Each object (instance) will have its separate data (E.g. a player1 will have their own name, score, etc.)
    - Each object (instance) will have a common set of actions (E.g. each player object can run, jump, etc.)

- Remember: Using OO is a choice, not a must but, it's a very popular choice and therefore it's a must to learn.

-

But let's write a non-OO program first

Scenario: Create a menu-driven calculator that can take in two numbers and perform +, -, * or / operations on the two numbers.

When Daryl demonstrates the code, note the following:

- The variables used and the inputs
- Sequence of the program

The Same Scenario (in English) and it's OO Design

- Create a menu-driven calculator that can take in two numbers and perform +, -, * or / operations on the two numbers.
  - To identify classes/objects ask:
  - Who or what are the entity types here? Class names are typically singular (why?)
  - To identify object/instance properties, ask:
  - What data will I need to hold for the entire duration of the object? (which variables shouldn't be in this list?)
  - To identify methods of a class, ask:
  - What actions will each class need to be able to perform?
  - Every program starts from somewhere, where will it start from?

Now let's write the OO version of the Java code

- When Daryl demonstrates the code, note how, when and where to:
  - Create new classes in the Eclipse project.
  - Create the constructor
  - Make the whole program start
  - Create new object/instance variables
    - Why private? Why no static?
  - Create new methods.
    - Why public? Why no static?
  - Create objects and access their variables, methods.

Note: This is just a teaser of OO. There are two more lectures of OO concepts and depending on your enrollment, you might have one or more courses on OO concepts. The code shown is simplified to make the transition from monolithic programming to OO, easier.

Optional reading: https://docs.oracle.com/javase/tutorial/java/javaOO/index.html