# COMP1007 Imperative Programming Part III

## Agenda

- More control statements.
- Compound statements
- Introduction to Scope and Lifetime
- Input

## Switch statement

- Select one int/char from many

```
switch (value)
{
        case 1 :
           c = ‘1’ ; break ;
        case 2 :
           c = ‘2’ ; break ;
        default :
           c = ‘?’
}
```

## break

- The break statement allows you to directly exit a loop.

```
while (true)
{
   x++ ;
   if (x > 10)
   {
      break ;    // When x > 10, exit loop and carry on
   }
}
```

## continue

- The continue statement allows you to jump to the next iteration of a loop.

```
while (true)
{
   x++ ;
   if (x < 4)
   {
     continue ;    // When x < 4, jump to next iteration
   }
}
```

## Comments

- // A one line comment
- /*
  Multi
  Line
  Comment
  */
- /** Documentation comments */

## The Compound Statement

- All the loop bodies have statements bracketed by braces: { }
- This kind of bracketed sequence of statements is called a *compound statement*.
- A sequence of statements is a kind of statement!

## Combining statements

- Any kind of statement, including a compound statement, can be used where a statement is expected.

```
while (x++ < 10)
{
   if (x == 2)
    {
       System.out.println("It's 2!") ;
    }
}
```

## Compound Statements and Variables

- A variable can be declared inside a compound statement:

```
{
   int x = 10 ;  // x is a local variable
   // x can now be used
   …
}
```

## Scope

- A compound statement defines a *local scope*.
- A *local variable* is only valid inside its local scope.

## Scope (2)

```
while (boolean-exp)
{
  int x = 10 ;
  // Use x here
}
// x not valid here
```

## Scope (3)

```
while (boolean-exp)
{
  int x = 10 ;
  // Use x here
}
while (boolean-exp)
{
  int x = 10 ; // different x from the one above
  // use x
}
```

Disjoint scopes

## Lifetime

```
while (boolean-exp)
{
   int x = 10 ;
   // Use x here
}
```

Local scope defines the lifetime of a local variable.

- x only exists when the scope is active (i.e., the program is executing the compound statement.)
- x is created and initialised every time the compound statement is executed.
- Every time the loop body is executed.

## Scope and Lifetime

- Important ideas.
- We will revisit them when we look at writing methods.

## Input

- Data read by a program.
  - Output is data written by a program.
- Input can come from the keyboard,
- Or from a data file,
- Or from a network connection.

## Input is awkward!!

- The supplied data can be of the wrong kind,
- Or the wrong value.

- Users typing at the keyboard make mistakes,
- And are often simply difficult!

## Reading from the keyboard

- Use the KeyboardInput class
  - See 1007 web page or the text book
- Provides an input object that can read:
  - int
  - double
  - char
  - String
  - And several other types.

## Reading from the keyboard (2)

- Use KeyboardInput like this:

```
KeyboardInput in = new KeyboardInput() ;
System.out.print("Type an integer: ") ;
int n = in.readInteger() ;
…
System.out.println("Integer was: " + n) ;
```

## readInteger

- Attempts to convert what the user types into an int.
  - I.e., user types “123”, giving the int 123.
- However, if the characters cannot be interpreted as an int, zero is returned.
  - I.e., user types “hello”, giving the int 0.

## Zero or error?

- How do you know if the user typed 0 or gave invalid input?
- You don’t!
- KeyboardInput objects are useful for learning to program but are not suitable for “real programs”.

## Interactive Programs

- Ask the user for input, then do something with the data.
- For example, ask for an integer and output the square of the integer.
- More interesting programs!

## Example Interactive Program

```
public class Program2
{
    public void run()
    {
      KeyboardInput in = new KeyboardInput ();
      System.out.print("Type your name: ");
      String name = in.readString();
      System.out.print("Type in a message: ");
      String message = in.readString();
      System.out.println("\n\nHello, you are: " + name);
      System.out.println("And your message is: " + message);
    }
  public static void main(final String[] args)
  { // etc.}
```

## Getting the correct input

- If the input is wrong the program can repeat the input code until it is correct.
- Put the input statement(s) in a loop.

## Getting the correct input (2)

```
KeyboardInput in = new KeyboardInput();
int n = 0;
do
{
  System.out.print("Type 0 to stop, 1 to continue:”);
  n = in.readInteger();
} while (n != 0 && n != 1);
```

## More Info?

- See the KeyboardInput web page.
- Do exercises 3.
- Read the text book!

## Summary

- Switch, break and continue.
- A compound defines a local scope.
  - Local variables are declared in a local scope.
  - The lifetime of a local variable is determined by its scope.
- Input enables interactive programs.