Introduction to Programming Review – what we covered

Module Learning Outcomes – what you should have learned to do

- At the end of this module the student will be able to:
  - L1. write small structured programs in a high level programming language
  - L2. demonstrate the use of standard programming constructs for iteration, selection and data structures such as arrays
  - L3. create a simple console-based user-interface for a program and use this to create interactive software

from the module descriptor

L1 Write small structured programs...

- Small?
  - But not (just) tiny!
  - A few hundred lines of code
- Structured?
  - Not monolithic (in Java terms, not having all the code in one subroutine/method)
  - Functionality distributed across a number of subroutines (methods) and components (classes)
  - Subroutines and components obey the rules of structured programming

L1 ...in a high level programming language

- In our case, Java
  - So have examined Java's programming constructs
    - Primitive types: eight of them, numeric and boolean values
    - Classes and their uses: predefined classes such as String and Exception, programmer-defined classes such as TextIO (utilities for I/O) and Student (whose instances are objects)
    - Variables: local, instance, and class (static) variables
    - Control flow: blocks, selection statements (if, switch), iteration (for loops, while loops, do-while loops); method calls and the main() method as the program's entry point; try-catch statements and exceptions (see L2)
    - Subroutines (static methods): parameters and arguments, return type
    - Data structures: "records" , arrays and ArrayList (see L2)

L2 demonstrate the use of standard programming constructs for iteration, selection….

- Use iteration
  - Definite loops
    - When you know in advance how many iterations there will be (for loop, enhanced for (aka for-each) loop)
  - Indefinite loops
    - When you do not know in advance how many iterations but continue while a condition is true; loop eventually changes the condition to false – or loop never stops! (while, do-while)
- Use selection
  - To decide whether an action is appropriate
    - if statement - depending on a condition
  - To choose which of a set of possible actions is appropriate
    - if-else statement, switch statement - depending on a set of conditions or on what value something has

L2 demonstrate the use of …. data structures such as arrays

- Use arrays to store a collection of elements of the same type
  - Fixed capacity or size (`array.length` in Java), set when the array object is created
  - Can think of an array as a collection of variables of the same type, accessed by an index (an integer)
  - May need to maintain a counter of the number of array positions that are actually storing a value
    - In this case, arrange that the empty (unused) elements are located at the end of the array
  - Can use `java.util.ArrayList` to avoid issues of fixed size and the need to maintain a counter

L3. create a simple console-based user-interface for a program and use this to create interactive software

- Display a menu of options on the console and ask the user to select one
  - We have used TextIO to display the menu and to get the user input
- Typically
  - Use a switch statement to process the user response and select and invoke the appropriate action
  - Use a do-while loop to repeatedly display the menu interface until the user elects to quit

Week 1 Topics

- Generalised model of a computer system
  - General purpose (e.g. PC) or Specialised (e.g. MP3 player)
  - Operating System’s role
  - Model a system using: Input **à** Process **à** Output
- Computer Programs
  - Machine code and assembly language
    - low level languages
  - High level languages
    - Translation by compiler versus translation and execution by an interpreter

Week 1 Topics continued

- Separating Data (things) and Process (logic)
  - Structured programming
- First Java application
  - Converting Fahrenheit to Celsius
- Handling Complexity
  - Decomposing the system into simpler parts
  - Combining the parts to form the final system

Week 2 Topics

- Programs, Algorithms and Data Types
  - Console applications in Java
    - Console input and output
    - The main() method
  - First look at concept of *algorithm*
    - A set of step by step instructions to solving a problem or performing a task
  - Instructions in Java
    - Statements and sequences - blocks

Week 2 Topics continued – data types

- A type is
  - a set of values
  - a set of operations that can be applied to those values
- Java's eight primitive types
  - Used to represent numbers, characters (letters), and true/false values
  - Type names are used in variable declarations, *e.g.*:

```
        double radius;
```

Week 2 Topics continued

- Variables
  - Have a type
  - Store a value of that type
- Local variables
  - Declared inside a block
    - E.g. in the body of a method such as main()
  - Must be *initialised* before they are used
  - Can use an *assignment statement* to give a variable its initial value, or to update (replace) the value that it stores

```
        radius = 1.234;
```

Week 3 Topics

- Values, Operators, Expressions and Statements
  - Primitive type values
  - An expression
    - is a construct made up of literals, variables, operators and method calls (where the method call returns a value)
    - evaluates to a single value
  - An operator
    - is a symbol such as "+", "<=", or "++" that represents an operation that can be applied to one or more values in an expression
  - Converting values of one primitive numeric type to another primitive numeric type
    - Automatic widening conversions, e.g. **int** to **double**
    - *Type casts* needed for narrowing conversions, e.g. **double** to **int**

```
          (int)(Math.random()*6) + 1
```

  - Simple statements, end with a semi-colon
    - Roughly equivalent to a sentence in a natural language

Week 3 Topics continued

- Classes as types
- String as an example class
  - String objects are immutable
  - String objects have methods that return values to manipulate them, similar to the operators on primitive types, e.g.
    - charAt(), length(), toUpperCase(), toLowerCase(), indexOf()
- Array and “record” classes to represent composite data

Week 4 Topics

- Control Structures 1 – Sequence and Selection
  - Structured programming
    - Arose in response to problems with the use of goto statement
    - Sequence, selection and iteration used to describe flow of control
  - Sequence
    - Statements in a block executed in sequence from top to bottom in the absence of any other control structure

Week 4 Topics continued

- Selection
  - Provides for control flow in which the program chooses which of a set of alternative actions it should perform based on some condition/value
- Selection in Java
  - if, if-else, if-else-if statements - actions based on true/false (boolean) values
    - Using blocks to group statements for each condition
    - Nested if statements and the “dangling else” problem
    - Compound boolean conditions and short-circuit operators
  - switch statement - actions based on integer, character, enum or String values
    - The issue of fall-through

Week 5 Topics

- Control Structures 2 – Iteration
  - Iteration
    - provides for control flow in which the program executes the same instructions over and over again
  - Iteration in Java: loop statements
    - for loop, while loop, do-while loop
    - Definite versus indefinite loops
      - Counting (for) and conditional (while, do-while) loops
      - Exiting a loop in the middle with a break statement (N½ loops)

Week 6 Topics

- Control Structures 3 – Exceptions
  - Dealing with errors and the unexpected
  - An exception
    - signals that an unexpected event has occurred that disrupts the normal flow of control of the program
    - normal flow of control is abandoned – control passes to an exception handler if there is one
    - allows the logic that deals with exceptional events to be separated from the logic that is normally followed

Week 6 Topics continued

- RuntimeException
  - Common runtime errors thrown by the runtime system
    - StringIndexOutOfBoundsException, NullPointerException
- try-catch statement
  - Try part contains statements that might throw an exception
  - Catch part(s) names an exception and contains code to be executed if that exception is thrown in the try part

Week 6 Topics continued

- A first look at arrays in Java
  - Array types, declared using [] after element type name
  - Array variables – can refer to array objects
  - Arrays as objects
    - Elements of same type, given a default value when the object is created
    - Fixed size, can use length field to interrogate this
  - Creating an array object with an array constructor
  - Indexing the array to get access to the individual elements
    - Index range is 0 to array.length-1

Week 7 Topics

- Static Methods (Subroutines)
  - A method is a sequence of statements with a name and zero or more parameters
    - The statements are grouped in a block and can include declarations of local variables as well as executable statements such as if statements and while loops
    - The statements are executed whenever the method is *called*
    - The name of the method is used to call it
      - Values must be supplied for any parameters, in parentheses, when the method is called

Week 7 Topics continued

- Static Methods
  - Belong to the class, can be called by giving name of class followed by name of method and any parameters. E.g.

```
double d = Math.random();
TextIO.putln(d);
```

    - As opposed to *instance methods*, which belong to an instance of a class and are called by giving the name of the instance followed by the name of the method and any parameters. E.g.

```
s = s.toUpperCase();  // s is a String
```

Week 7 Topics continued

- Method declarations
  - Includes signature
    - Method name and list of parameter types
  - Return type
    - The type of value the method returns, or **void** if none
    - Not part of signature
  - Each method in a class must have a unique signature
    - Methods can have same name if parameter list is different (overloading)

Week 7 Topics continued

- Method body
  - Contains the code that is executed when method is called
    - Is never executed if method is never called
  - Details not visible to the calling code
    - Simplifies program understanding (“what” not “how”)
    - Details can be changed without affecting calling code
- Calling a method results in the method body executing (control transfers to method body)
  - When this has finished control returns to the caller and execution continues with the statement following the method call

Week 7 Topics continued

- Method parameters
  - Method declaration names *formal parameters*
  - Method call must supply an *actual parameter* for each formal parameter
  - In Java, the value of the actual parameter is copied into the formal parameter
    - So method body uses a copy of the value supplied as the actual parameter
    - Formal parameter is a local variable, distinct from any variable supplied as actual parameter

Week 8 Topics

- Static Methods part 2
  - Methods as named actions
  - Method contracts
    - What the caller needs to know to call the method
    - Preconditions (what the method requires) and postconditions (what the method will do if its requirements are met when it is called)
    - Avoiding preconditions (minimising a methods requirements)
  - Using static variables and side-effects

Week 9 Topics

- Developing an Algorithm
  - Formalizing the algorithm using flowcharts and program design languages
    - Stepwise refinement of steps in both approaches
  - Documenting assumptions, effects, inputs and outputs
    - Javadoc comments

Week 9 Topics continued

- An example algorithm
  - Bubble sort
    - Trade-offs between ease of writing and how well the algorithm performs
      - Bubble sort easy to write but does not perform well
    - Flowchart description
    - PDL description
    - Java specification (method “header”)
    - Implementation (method body)

Weeks 11 and 12

- Lectures covered the practice programming projects for last session
  - Puzzle of mama, baby, crab and cone
- and this session
  - Towers of Hanoi with 3 disks

T2 Weeks 1, 2 and 3 Topics

- A more detailed introduction to arrays in Java
  - ArrayIndexOutOfBoundsException
  - Diagrammatic representations of array
  - Arrays with more than one dimension
  - Aggregate assignment to an array using a list of element values
  - Iterating over an array with a for loop or a for-each loop
  - Arrays and methods – methods that return an array

T2 Weeks 1, 2 and 3 Topics continued

- Iterating over an array in reverse
- Var-arg methods

```
public static double average(double… numbers)
```

  - Above method can be called with any number of double arguments, separated by commas
  - List is copied into an array when the method is called
- ArrayList
  - Use wrapper classes for ArrayList objects whose elements are primitive type values (e.g. ArrayList<Integer> for int)

T2 Week 4 – Example program

- Lecture reviewed the solution to the Towers of Hanoi practice practice project with 3 disks then extended example to deal with any number of disks
  - Intended to illustrate use of most of programming concepts and structures covered in module to that point
  - Intended to help prepare you for the real programming project

T2 Week 5 Topics

- Types, Classes and Objects – part 1
  - Classes as applications
  - Classes as containers for static methods
    - E.g. Math, TextIO
  - Classes as types
    - Define a set of values and a set of operations on those values
      - E.g. String

T2 Week 5 Topics cont'd

- Classes as reference types
  - Values are references to instances of the class (objects)
- Objects have
  - State (recorded in instance variables)
  - Behaviour (initiated by calling the object's methods)
- Objects are created by calling a *constructor*
  - Creates and initialises the instance
  - Returns a reference to the instance

T2 Week 5 Topics continued

- The value, **null**
  - A reference type variable contains a reference to an instance of the type, or has the value **null** (does not refer to any instance)
- Equality for reference types
  - Compares whether references are to the same instance
    - Can compare a reference to **null**
- Assignment for reference types
  - Copies the reference to an instance, not the instance itself
    - Can assign the value **null** to a reference variable

T2 Week 5 Topics continued

- Declaring instance variables as `private`
- Getters and Setters
  - Allow class to control access to the state of its instances
    - Getters return value, if no getter value cannot be accesses
  - Allow class to ensure that updates that change the state of an instance are valid so that the state remains consistent
    - Setters allow updates to values of instance variables, but can include logic to prevent invalid updates

T2 Week 6 Topics

- Object-based (and object-oriented) programming
  - Combine data and processing code, defining both in same place (encapsulation)
  - In Java these are encapsulated in a class declaration
    - Instances of a class are *objects*
    - Objects have *state* and *behaviour*

T2 Week 6 Topics continued

- The class Object and its methods
  - All classes inherit methods of Object
  - These include: toString() and equals()
- Redefining the toString() method of a class so that it returns a useful String representation of the state of an instance of the class
  - TextIO.put() and System.out.print() call the toString() method of an object to display it

T2 Week 7 Topics

- This session's programming project was discussed in the lecture
- The open Knight's tour (or Knight's path)
  - A program to allow the user to move the knight around the board to find a tour
  - More marks for well-structured solution
  - More marks for additional features to the minimal set asked for

T2 Week 8 Topics

- TextIO and Files
- Using TextIO to create and read files of text
  - Changing the input for TextIO using readFile()
  - Changing the output for TextIO using writeFile()
  - Closing the file and returning the input to the keyboard using readStandardInput()
  - Closing the file and returning the output to the console using writeStandardOutput()

Next week

- No lecture
- Practice class test in the lab
  - 20 multiple choice questions, same format as before, but on the whole course
  - Just a practice test, but should help you judge how much work you need to do for the real test, which is the following week

Final Week (T2 Week 12)

- Again, no lecture
- Class Test 2 in the lab
  - Counts for 10% of module marks
  - Same format and range of topics as the practice test next week
- Programming Project
  - Due on the Friday of that week