COMP103P Object-Oriented Programming
Programming Notes and Exercises 1
Finish by Friday 22nd January 2016

Purpose: Some practice of writing programs in Java that use methods and files. The challenge is to learn some Java and get programming quickly!

Goal: Complete as many of the exercise questions as you can. If you are keeping up, you need to do at least the core questions. The additional questions are more challenging and are designed to stretch the more confident programmers. If you can't do them now, be prepared to come back and try them later on. You must complete the core questions and get them reviewed in a lab session.

Feedback: It is important that you get feedback on your exercise answers so that you know they are correct, that you are not making common mistakes, that the program code is properly presented and that you are confident you have solved the problem properly. To do this, get your answers reviewed by a lab demonstrator during lab sessions.

See the additional notes on the COMP103P Moodle site about using Java, and classes Input, FileInput and FileOutput.

Using An IDE

You need to learn how to use the an IDE efficiently and the best way to do that is to use it a lot! All the exercises should be done using an IDE. Eclipse is installed on lab machines. Both Eclipse and IntelliJ IDEA can be installed on your own machine.

Don't forget that for each program you write you should create a new Eclipse or IDEA project.

Example Questions and Answers — READ THESE CAREFULLY

The following are examples of questions and answers, to show how the Java syntax works, and to illustrate the kinds of programs you should be writing.

Example 1.1 Write a class that has the following methods:

- public int sumOfDigits(int n) — to find the sum of the integer parameter's digits and return the result.
- public void inputAndProcess(Input in) — to input an integer, call sumOfDigits and display the result.

Hint: % is the division operator that returns the remainder. Try using it to divide by 10.

Answer:

```
// Written by A.Person, January 2016
// Answer to Exercises 1 example question 1
class Example1_1
{
  public int sumOfDigits(int n)
  {
    int sum = 0;
    n = Math.abs(n);
    while (n > 0)
```

```
    {
      sum += n % 10;
      n /= 10;
    }
    return sum;
  }

  public void inputAndProcess()
  {
    Input in = new Input()
    System.out.print("Type an integer: ");
    int n = in.nextInt();
    System.out.print("The sum of the digits of: " + n);
    System.out.println(" is: " + sumOfDigits(n));
  }

  // The main method should do no more than create the object
  // and call a method to do the work.
  public static void main(String[] args)
  {
    new Example1_1().inputAndProcess();
  }
}
```

This uses the Input class to read from the keyboard via the terminal window. To get a copy of the Input class and information about how to use it, see the notes on the Moodle web site. Class Input is very strict about input being entered correctly, so will terminate the program if the input is incorrect. For example, if the user types in some letters rather than valid digits, the program will display the message 'Input did not represent an int value.' and terminate.

To avoid the program terminating, a better version of the inputAndProcess method would be this:

```
public void inputAndProcess()
  {
    int n = 0;
    Input in = new Input();
    while (true)
    {
      System.out.print("Type an integer: ");
      if (in.hasNextInt())
      {
        n = in.nextInt();
        break;
      }
      in.nextLine();
      System.out.println("You did not type an integer, try again.");
    }
    System.out.print("The sum of the digits of: " + n);
    System.out.println(" is: " + sumOfDigits(n));
  }
```

A loop is used to repeatedly ask for input until something that can be recognised as an integer is entered.

An alternative to reading input from the terminal window is to use the GUIInput class that displays a simple dialog box into which input can be typed. You can get a copy of the source code of the class from Moodle as the file GUIInput.java. A copy of the GUIInput.java file should be included in the same Eclipse project as the example class (create a new class in Eclipse and copy and paste the code into the editor) and the inputAndProcess method changed to this:

```
public void inputAndProcess()
{
  GUIInput input = new GUIInput();
  int n = input.readInt("Type an integer: ");
  System.out.print("The sum of the digits of: " + n);
  System.out.println(" is: " + sumOfDigits(n));
}
```

The GUIInput class provides the following methods for input:

```
public int readInt(String prompt)
public long readLong(String prompt)
public double readDouble(String prompt)
public float readFloat(String prompt)
public String readString(String prompt)
public char readChar(String prompt)
```

In each case the parameter called prompt is the prompt to display in the dialog telling the user what to type in. If the user types in the wrong kind of value, the dialog will be redisplayed until the correct kind of value is entered. If the Cancel button is pressed a default value is returned, zero for the numerical types, an empty string or a space character.

Further thoughts:

- Does the program work correctly with negative integers? Copy, compile and run the program to find out.
- Why is the Input object created in the main method and not the inputAndProcess method? Is this a good decision?
- Do you think that the inputAndProcess method is a good *cohesive* method? The 'and' in the method name is a clue! How might you change the method?
- Note the use of Math.abs. This is a function implemented as a utility method in class Math (see the online class documentation for more details: http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html). The abs function returns the absolute value of a double, basically meaning that it removes the minus sign from any negative number.

Example 1.2 Write a one-class program that has a method to ask for and return the name of a text file, a method taking a file name (a String) as a parameter that reads and displays each line of text in the file on the screen, and a method to call the first two methods.

```
// Written by A.Person, January 2016
// Answer to Exercises 1 example question 2
class Example1_2
{
  private void displayFileContent(String filename)
  {
    FileInput fileIn = new FileInput(filename);
    while (fileIn.hasNextLine())
    {
      String s = fileIn.nextLine();
      System.out.println(s);
    }
    fileIn.close(); // Always close a file after it has been used.
  }

  private String getFileName()
```

```
  {
    Input in = new Input();
    System.out.print("Enter filename: ");
    String filename = in.nextLine();
    return filename;
  }

  public void showFile()
  {
    String filename = getFileName();
    displayFileContent(filename);
    // Could have written displayFileContent(getFileName());
  }

  public static void main(String[] args)
  {
    new Example1_2().showFile();
  }
}
```

This uses the FileInput class. To get a copy of the class and information about how to use it, see the notes on the Moodle web site. Look at the way that data is read from the file in the while loop. The hasNextLine() method will return true if there is more data to read from the file. If it returns false the file copy is finished.

Note that the methods displayFileContent and getFileName have been made private. The public showFile method, acting as a controller, is the only one that can be called for an object in the main method.

Example 1.3 Write a one-class program that uses a method to display a multiplication table, given an integer between 2 and 12 inclusive. The program should ask the user which table to display and reject any invalid request.

There will be a method that displays the multiplication table, which can be called displayTable. This will take an integer parameter but does not need to return a result, so can be declared as void displayTable(int n). Also there will be a main method, as all programs must have one. However, should the main method ask the user for input and call displayTable, or should that behaviour be put into another method?

The answer to this question lies in an important method design principle: each method should be cohesive — that is, it should focus on doing one, and only one, well-defined chunk of behaviour needed by the program. Following this principle, the main method can focus on initialising the program by creating an object and then call another method to handle the user input. That method can, in turn, then focus on doing one thing before calling the method to display the table. Hence, we now have a method called doTable, and established which method calls another.

With the methods and their behaviour identified, each can now be looked at in turn to design their respective method bodies. Moreover, rather than having to consider the entire statement sequence in the program at one go, we are now concerned only with small portions at a time. This has divided a larger and more complex design problem into a collection of smaller, and more easily designed, components. Further each component has a well-defined interface by which it connects to other components. The interface is the name of the method and its parameter list (i.e., the number and types of the parameters) and the returned value if not void.

The design of each method is now straightforward. Method displayTable needs a loop and output statements to display a multiplication table, while doTable asks the user to enter a value and checks it is within the supported range. If it is then displayTable is called, otherwise an error message is displayed.

```
// Written by A.Person, January 2016
// Answer to Exercises 1 example question 3
// Program to display a multiplication table.
public class MultiplicationTable
{
  // Display multiplication table for n.
  private void displayTable(final int n)
  {
    int counter = 1;
    System.out.println("The " + n + " times table");
    while (counter < 13)
    {
      System.out.print(counter + " x " + n);
      System.out.println(" = " + counter * n);
      counter = counter + 1;
    }
  }

  // Input table to be displayed and get it displayed.
  public void doTable()
  {
    Input in = new Input ();
    System.out.print("Which table (2-12)? ");
    int x = in.nextInt();
    if ((x < 2) || (x > 12))
    {
      System.out.println("Cannot display that table");
    }
    else
    {
      displayTable(x);
    }
  }

  public static void main(final String[] args)
  {
    new MultiplicationTable().doTable();
  }
}
```

Further thoughts:

- Note where the Input object is created in this program. Is this the correct decision? The strategy should be to always minimise the scope within which a name is visible.
- Is the method doTable actually well-designed and cohesive? Perhaps it would be better to change it so that it only does the input and returns an integer, removing the if statement to a new method.

A method name should be chosen to help describe what the method does. Are the names chosen in these example programs good enough or can you suggest better names? Does doTable properly suggest the role of the method?

Example 1.4 Write a class that has methods to input a collection of integers, storing them as integer objects in an ArrayList<Integer>, to sort the ArrayList<Integer> and to display the sorted contents of the ArrayList<Integer>. The ArrayList<Integer> should be stored using an instance variable.

```
// Written by A.Person, January 2016
// Answer to Exercises 1 example question 4

import java.util.ArrayList;   // Need to import the class before it
                              // can be used.
import java.util.Collections; // Class Collections provides a sort
                              // method for Collection objects
                              // such as ArrayLists.

class Example1_4
{
  private ArrayList<Integer> numbers = new ArrayList<Integer>();

  private void displayIntegers()
  {
    for (int i : numbers)
    {
      System.out.println(i);
    }
  }

  private void inputIntegers()
  {
    Input in = new Input();
    System.out.print("How many integers to input? ");
    int count = in.nextInt();
    for (int n = 0; n < count; n++)
    {
      System.out.print("Enter integer (" + n + ") : ");
      int value = in.nextInt();
      numbers.add(value);
    }
  }

  private void sortIntegers()
  {
    // Easy if you can find and use a method in the class library!!
    Collections.sort(numbers);
  }

  public void run()
  {
    inputIntegers();
    sortIntegers();
    displayIntegers();
  }

  public static void main(String[] args)
  {
    new Example1_4().run();
  }
}
```

This example makes use of the generic ArrayList class, where the <Integer> notation denotes that the ArrayList can hold Integer objects only. Class Integer is a library class that represents int values

as objects, in contrast to primitive values of type int, which are not objects. A mechanism called auto boxing and unboxing automatically deals with conversions between int values and Integer objects, allowing int values to be stored in an ArrayList<Integer> without needing to do any conversions explicitly.

Notice how easy the sorting turned out to be! A suitable sort method already exists in the class library, the only work lies in finding the class and method — use the index in the Java class documentation. Also notice the loop used to print the ArrayList<Integer>:

```
for (int i : numbers)
 {
   System.out.println(i);
 }
```

This is using an enhanced for loop to access each element of the ArrayList<Integer> in turn. Auto-unboxing automatically takes care of converting from the Integer objects stored in the ArrayList<Integer> to the int values used in the loop.

Example 1.5 Write a one-class program that determines whether a word or sentence is a palindrome. A palindrome reads backwards the same way as forwards, for example: "Never odd or even". Spaces and lower/upper case are not considered as significant.

Notes: The problem statement gives little information about designing the program, so we definitely need to do some thinking and planning before trying to write code. First of all we need an algorithm to test whether a string is a palindrome. This turns out to be easy:

1. make a copy of the string

2. reverse the order of the characters in the copy

3. compare the reversed copy with the original to see if they are the same

Now that we know that it is possible to check for a palindrome, the next step is to consider the overall sequence of events that should take place when the program runs. This gives:

1. input string to test

2. make a copy

3. do the reverse

4. check the reverse copy against the original

5. display the answer

Next we want to determine what methods are required and allocate the steps listed above to them. We could bundle everything into a single method but that would violate our cohesion guidelines. Instead, we identify the following methods:

```
String getInput() // Get the input string to be tested. (1)
String reverse(String s) // Return the reverse of the argument String
                         // (2,3)
boolean check(String s1, String s2) // Is s2 the reverse of s1? (4)
void testForPalindrome(String s) // Call the check method and display
                                 // the answer (5)
```

This may seem more methods than necessary. That could turn out to be true but at this stage we are deliberately breaking the overall behaviour down into minimally sized methods to get a good feel for the design. Methods can be combined and behaviour re-allocated later if it seems appropriate. Note that making a copy of a String is a basic operation that does not need to put into a separate method on its own.

With the methods identified, each can be considered in turn to design its method body. Method getInput is easy as it simply requests input and returns the resulting string. TestForPalindrome is also straightforward, it just need to call the check method with the same string for both parameter values and display a suitable message giving the result. Check is a bit more interesting as the decision has been made to make it more general purpose than is strictly necessary for this version of the program. The idea for check is that it can actually test any two strings to see if they are the reverse of each other. TestForPalindrome will call it with the same string for both arguments since it checks for self-reversed strings. Check works by reversing one string and then doing a string comparison using the compareTo method in the String class.

This leaves reverse, which requires a bit more work. The basic strategy here is to unpack the original string character by character and reassemble the characters in the reverse order to create a new string. To find out how to extract individual characters from a string we need to return the documentation for class String. This shows that there is a method declared as char charAt(int index), which returns the character at position index, counting from 0 as the position of the first character. The String documentation also shows a method int length() that returns the number of characters in a String. With these methods we can construct a loop to access each character in turn:

```
// Given some String s
int position = 0;
while (position < s.length())
{

  // Characters run from 0 to length-1
  ...
  s.charAt(position); // Get the character and do something with it
  ...
  position = position + 1; // Move to next character
}
```

Building the new string requires a search of the class documentation to find a method to join a char to a String. The String class does not provide one, so we start looking at other classes. It turns out there is a class Character, whose objects represent characters, that can be used to create a String from a single char using an expression like:

```
// Given a char c
... new Character(c).toString() ...
```

The toString() method converts from a Character object to a String object. The two strings can then be concatenated using the + operator.

We now have all the pieces ready to write the program. However, a review of the problem statement to check we are still answering the right question highlights a problem: the sample palindrome is only a palindrome if the case of the letters is ignored and the spaces are ignored. If the case of the letters or position of spaces is significant then the sample string is not a palindrome. If our current design was applied to this sample string, it would not be accepted as a palindrome. This can be resolved by providing a helper method called tidyString that takes a String as a parameter and returns a String that is a copy without spaces and all letters in lowercase.

Before writing the tidyString helper method, a further search of the String class documentation is a good idea to see if there are existing library methods that can do the work. This reveals the String toLowerCase() method, which returns a lower case version of the string it is called for, and the String replaceAll(String target, String replacement) method, which will replace all occurrences of the String target with the String replacement.

```
// Written by A.Person, January 2015
// Answer to Exercises 1 example question 5
// Program to check for a palindrome.
public class Palindrome
{
```

```
// Return a String which is the reverse of the argument String
private String reverse(final String s)
{
  String result = new String();
  int position = 0;
  while (position < s.length())
  {
    result = new Character(s.charAt(position)).toString() + result;
    position = position + 1;
  }
  return result;
}

// Check to see if the two argument Strings are the reverse of each
// other. Return true if they are and false otherwise..
private boolean check(final String s1, final String s2)
 {
  String s = reverse(s2);
  if (s1.compareTo(s) == 0) {
    return true;
  }
  else
  {
    return false;
  }
}

// Get user to input a String.
private String getInput()
{
  Input in = new Input ();
  System.out.print("Enter text to check: ");
  return in.nextLine();
}

private String tidyString(final String s)
{
  return s.replaceAll(" ","").toLowerCase();
}

// Do the palindrome test and display the result
public void testForPalindrome(final String s)
{
  if (check(tidyString(s), tidyString(s)))
  {
    System.out.println("String is a palindrome.");
  }
  else
  {
    System.out.println("String is not a palindrome.");
  }
}

public void go()
{
  testForPalindrome(getInput());
```

```
  }

  public static void main(final String[] args)
  {
    new Q10().go();
  }
}
```

Note that all the method parameters have been declared as final. This is an idiom that some programmers like to use. The value of a final parameter cannot be altered within the method body, so declaring a parameter final lets the programmer explicitly state that the parameter should not be changed and get the compiler to check that it doesn't. When a variable of class type is declared final, it cannot be changed to refer to a different object. However, the object being referenced can still be altered by calling a method that changes its state (i.e., the value of an instance variable held by the object is changed).

Further thoughts:

- Should the getInput method be called in the main method?
- Is there a method in the Java class libraries that can reverse a String to avoid having to write a method?

Core questions

You should have already written answers to some of these questions in C. Make sure you can also answer them using Java.

Don’t forget to pay attention to how your source code is formatted, and keep your methods short and cohesive. Practice using the features in your IDE, in particular editor short cuts, code-completion and debugging. Don’t forget that you create a new IDE project for each program you write.

Q1.1 Write a program that inputs a sequence of Strings until the word **stop** is entered.

Hint: You compare strings using compareTo, as outlined in the preceding notes.

Q1.2 Write a program to input 10 doubles (values of type double) and store them in an array. Then compute and display the average of the values input. Try entering negative as well as positive numbers. Consider writing one method to do the input and another to compute the average.

Q1.3 Write a program to input 10 words, storing them as strings in an ArrayList<String> (not an array), and then display the words in reverse sorted order.

Q1.4 Write a program to generate 10000 random doubles between -0.9999999 and +0.9999999. Print out the largest, smallest and the average values.

Do you need an array? Think very carefully about this.

Hint: Look at the documentation for class Random and find the method nextDouble.
Another hint: Negative values?? Class Random also has nextBoolean...

Note: Answers to the following questions involve writing programs consisting of two or more methods (not including the main method).

Q1.5 Write a one-class program that has the following methods:

- a method that inputs and returns a double value,
- a method that takes two double parameters, adds them together and returns the square root of the result,
- a main method to create an object and call the other two methods, displaying the result of calling the second method.

Q1.6 Repeat Q1.5 but store the double values in two instance variables rather than using parameters.

Hint: you will need another instance method to call the input method to get values to store in each instance variable and to call the other methods. This instance method should be public, the others private. The input method should be unchanged.

Q1.7 Write a method: public String toBase(int n, int b) that converts n to its String representation in number base b, i.e., toBase(3,2) will return "11". Put the method into a one-class program that allows you to input values and use the method.

Q1.8 Write methods to do the following:

- Convert from millimetres to feet.
- Convert from metres to inches.
- Convert from kilometres to yards.

Include the methods in an interactive program that lets the user select which conversion to perform, and then inputs a value and displays the result.

An interactive program means one that displays a simple text menu, such as:

```
1. Convert millimetres to feet.
2. Convert metres to inches.
3. Convert kilometres to yards.
4. Quit
```

and then asks the user to type in a value from 1 to 4 to select which action to take. A loop will continually redisplay the menu until the user selects Quit (number 4) to terminate the program.

Notes: The conversion methods should take the value to be converted as a parameter and return a result. They should not do any input or display the result. Add additional methods if they will help structure the program.

Q1.9 Write a one-class program to determine if a long integer is a palindrome (i.e., represents the same value when reversed, for example 123454321).

Q1.10 The example palindrome program seen in the preceding notes includes a tidyString method to remove spaces and make all characters lower case before checking whether a sentence is a palindrome. However, many example palindromes also include punctuation and spaces, for example “Neil, a trap! Sid is part alien!”. Modify the example palindrome program to remove the punctuation and spaces and compare any string.

Q1.11 Write a one-class program with suitable methods to read a text file character by character and count how frequently each character occurs.

Hints: You will need to use the FileInput class, which is used in a similar way to the Input class. See the notes on using FileInput.

The FileInput .class file needs to be in the same directory as the program you are working on. Copy FileInput.class to all directories containing programs that need to use it.

Q1.12 Write a program using methods that copies and reverses a text file (i.e., so that the destination file contains a backwards copy of the original file contents).

This program will need both the FileInput and FileOutput classes (described in separate notes).

Q1.13 Further modify the palindrome program to read a file containing strings (one per line) and checking whether each string is a palindrome. The palindromes found should be written to a separate output file. Note, do a Google search to find example palindromes.

Q1.14 Write a program using methods to display your name, or any other message, in the *middle* of a line 80 characters wide.

Q1.15 Write a program that uses a *recursive* method to calculate the product of a sequence of numbers specified by the user. For example, if the user specifies 4 to 8, the method calculates 4*5*6*7*8. Any range can be used, including the use of negative numbers, and the program must correctly determine the values in the range. Note that Java does not support ranges directly like Groovy does.

Q1.16 Write a method that takes two character array parameters and returns true if both arrays contain the same characters but not necessarily in the same order. Note, character arrays are of type char[].

Harder Questions

Q1.17 Write a method to test if an integer of type long is a prime number. The method should return a boolean value. Test your method by writing a test one-class program that reads an integer typed at the keyboard and states whether the integer was prime.

Next, using your prime method, write a program that finds all the prime numbers that can be represented by an integer of type long.

Notes: This is not quite as easy as it might appear, especially if you want the program to produce results quickly. Search the web for information about prime numbers and algorithms for finding them - there are some excellent web sites.

Q1.18 Write a program that reads an integer between 0 and 999 and "verbalises it". For example, if the program is given 123 it would display "one hundred and twenty three".

Hint: Write methods to deal with ranges of numbers, such as single digits between "zero" and "nine", numbers between 10 and 19 and so on.

Challenges

Q1.19 Write a program that reads a text file containing Java source code and checks that there is a matching number of opening and closing braces (curly brackets). Beware of braces appearing in comments, or in character literals (strings) and constants.

Q1.20 Redo some of the earlier questions using one or more different Object-Oriented programming languages. In particular look at Groovy, C++, Ruby or Python. You will find plenty of information about these languages on the web.