PGCert IT: Programming for Industry

Control Flow

Git Repository

The Git remote repository for today's lab is located at this link. **Remember to FORK yourself a personal copy** of this repository – don't just clone it!

Exercise One: if statements on paper

Do the following **on paper**!

1. Write a Java `"if"` statement that prints out `"Healthy weight"` if the value of the variable, `bmi`, is between 19 and 25 (inclusive).
2. Write a line of Java code which declares a boolean variable named `hasFinished` with an initial value of false. Then, write some Java code which calls the `printResults()` method if the boolean variable `hasFinished` is true. You may assume that the `printResults()` method has been implemented elsewhere.

Exercise Two: boolean expressions on paper

Do the following **on paper**!

1. Write a Java `boolean` expression which tests whether the value of the `char` variable, `userResponse`, is equal to either `'y'` or `'Y'`.
2. Write a Java `boolean` expression which tests whether the value of the `int` variable, `amount`, is odd (i.e. not evenly divisible by 2).
3. Write a Java `boolean` expression which tests whether the `String` variable, `firstName`, begins with the letter `'A'` or `'a'`.
4. Write a Java `boolean` expression which tests whether the `String` variable, `singer`, is equal to `"Taylor Swift"`. (Hint: remember that `Strings` are objects, not primitive types.)
5. Write a Java `boolean` expression which tests whether the value of the `int` variable `yearBorn`, is greater than 1978 but is not equal to 2013.

Exercise Three: if … else if statements on paper

Do the following **on paper**!

Complete the `getGender()` method below so that it assigns the correct value to the `gender` variable according to the `code` passed in as a `char` parameter.

The gender will be determined as follows:

- If `code` is equal to `'F'` or `'f'` the method should assign `"Female"` to `gender`
- else if the `code` is equal to `'M'` or `'m'`, then the method should assign `"Male"` to `gender`
- else the method should assign `"Unknown"` to the `gender` variable.

```
private String getGender (char code) {
   String gender;
   // TODO write your code here



   return gender;
}
```

Exercise Four: while loops on paper

What is the output produced by the following code fragment? Do this exercise **on paper!**

```
int number = 5;
while (number < 15) {
      System.out.print (3 * number + "  ");
      number += 4;
}
System.out.println();
System.out.println("Number is now: " + number);
```

Exercise Five: for loops on paper

Do the following **on paper**!

Using a `for` loop, complete the `printRowOfAmpersands()` method so that it prints a row of ampersands (&). The number of ampersands it should print is passed via the `int` parameter, `howMany`.

For example, when called by: `printRowAmpersands(5);`
The method prints: &&&&&

```
private void printRowOfAmpersands (int howMany) {
	// Write your code here

	System.out.println();
}
```

Exercise Six: Converting a while loop into a for loop

Do the following **on paper**!

Translate the following **while** loop into a **for** loop.

```
int i = 0;
while (i < 7) {
	System.out.print(2 * i + 3);
	i++;
}
```

Exercise Seven: Guessing game using a while loop

Write a program so that the user can play the game of guessing a number between 1 and 100. Use the following pseudocode to write the code:

- Generate a random number between 1 and 100 and store in a variable named `goal`
- Declare a variable named `guess`
- Initialise `guess` to 0
- While the user's guess is not correct (i.e. `while guess != goal`):
    - Ask the user to enter their guess
    - Store the guess in the `guess` variable
    - If the `guess` is greater than the `goal`, print "Too high, try again"
    - Else if the `guess` is less than the `goal`, print "Too low, try again"
    - Else print the message "Perfect!!"
- Print "Goodbye"

Here is an example of the output of the game:

```
Enter your guess (1 - 100): 50
Too low, try again
Enter your guess (1 - 100): 75
```

```
Too high, try again
Enter your guess (1 – 100): 70
Perfect!
Goodbye
```

The skeleton code is found in:
`ictgradschool.industry.controlflow.guessing.GuessingGame.java`

Exercise Eight: CodeRunner Exercises

Use the skeleton code found in `ictgradschool.industry.controlflow.coderunner.CodeRunner.java` when answering the following questions:

1. Complete the method `areSameName()` such that:
    a. When given two names that match exactly, outputs `"Same name"` , otherwise
    b. When given two names that have the same first character (case sensitive), outputs `"Same first letter"`, otherwise
    c. Output `"No match"`

    Hint: remember you can nest and/or chain conditional flow structures together.

2. Complete the method `isALeapYear()` such that:
    a. If the given year is exactly divisible by 4 return true, unless
    b. The given year is also exactly divisible by 100 but not exactly divisible by 400, in which case return false, otherwise
    c. Return false

    Hint: if you're struggling look up leap year on Wikipedia and look for 'algorithm'.

3. Complete the method `reverseInt()` that takes an integer and outputs the integer that has the number in reverse order. If given a negative integer, the number should be the reversed digits with the negative sign staying at the front, i.e. `reverseInt(-45) => -54`.

    Do **not** use String conversion or methods – there is a mathematics answer for this one. Hint: you can find the value in the rightmost digit of a number (the 'ones' place) by using number modulo 10, and you'll need to loop through the input number.

4. Complete the method `reverseString()` that outputs the given string in reverse order. You may assume the string is non-empty.

5. Complete the method `simpleMultiplicationTable()` that, given an integer, outputs a table showing the multiplication table (rows and columns) starting at 1 and up to and including that number. Any 'cell' in the table should display the result of

multiplying that row number by that column number. So for an input of 2, I'd expect the method to output a table 2 columns wide and 2 rows tall like so:

```
1 2
2 4
```

Then an input of 3 would result in a 3x3 table like so:

```
1 2 3
2 4 6
3 6 9
```

Hint: remember, you can nest loops too! Also, `"\n"` is the special character for new line.

6. Complete the method `convertIntToColTitle()` that converts an integer to excel column letter; column 1 being "A", column 2 is "B", while column 27 is "AA". This time you also need to include some sanity checking to ensure the integer passed in is valid (must be greater than 0) and if not, output the string "Input is invalid". Hint: you can cast a char to an int and vice-versa, e.g. `(int)'A'` results in 65 and `(char)90` results in `'Z'`.

7. Complete the method `isPrime()` that outputs true if the given integer is a prime number, false otherwise. As a reminder, a prime number has only two exact divisors, 1 and itself. 1 is not prime. Hint: the largest potential divisor for a number X is X / 2. Also, return early. Bonus points – assuming you take the 'naïve' approach to solving this question, can you tell me what a potential drawback of your technique is?

8. Complete the method `isIntPalindrome()` which determines if the digits of the given integer are palindromic… reads the same backwards as forwards. Any leading negative signs should be ignored for the purposes of this exercise. Hint: have we written any methods that might help us?

9. Complete the method `isStringPalindrome()` which determines if the given string is a palindrome. Whitespace should be ignored. Assume all strings are non-empty and in lower case. Hint: if you got the hint above, then this question is almost too easy; only tricky bit is removing whitespace… lookup `String.replaceAll()`.

10. Complete the method `printPrimeNumbers()` that return a String containing a space separated list of all of the prime numbers starting at 2 and all the way up to and including the given integer. If no prime numbers are found, return "No prime number found". Hint: another question that a previous method can help with. Ensure there isn't an extra space at the end of the String returned.

(Challenge) Exercise: The game of rock, paper and scissors

This application allows the user to play the game, “Rock Paper Scissors”. This game is played as follows: in each round the player choose either Rock, Paper or Scissors, and the computer randomly chooses either Rock, Paper or Scissors. The winner of that round depends on the items chosen by the computer and the player. If the same item is chosen, the result is a tie. If the two chosen items are rock and scissors, then the rock wins, because a rock smashes scissors. If scissors and paper are chosen, the scissors win, because scissors cut paper. If paper and rock are chosen, the paper wins, because paper covers rock.

Here is an example of the output of the application:

```
Hi! What is your name? Yu-Cheng

1. Rock
2. Scissors
3. Paper
4. Quit
Enter choice: 1

Yu-Cheng chose rock.
Computer chose scissors.
Yu-Cheng wins because rock smashes scissors.

1. Rock
2. Scissors
3. Paper
4. Quit
Enter choice: 2

Yu-Cheng chose scissors.
Computer chose scissors.
No one wins.

1. Rock
2. Scissors
3. Paper
4. Quit
Enter choice: 3

Yu-Cheng chose paper.
Computer chose scissors.
```

```
The computer wins because scissors cut paper.

1. Rock
2. Scissors
3. Paper
4. Quit
Enter choice: 4

Goodbye Yu-Cheng. Thanks for playing :)
```

When testing the user choice or the computer choice, you must use the symbolic constants that have been declared at the top of the program. E.g.

```
public static final int ROCK = 1;
```

You will need to implement the following methods:

1. `displayPlayerChoice()` method

The `displayPlayerChoice()` method has one `String` parameter and one `int` parameter. Complete the method so that it prints one line of output. The first part of the output is obtained from the `String` parameter. The second part depends on the second `int` parameter (which you should compare with the global constants). Depending on what value the user has entered, the second part of the output will be either: " chose scissors", " chose rock", or " chose paper".

2. `userWins()` method

The `userWins()` method has two `int` parameters and returns a `boolean`. The first `int` parameter is the user's choice and the second int parameter is the computer's choice. The method returns true if the user choice beats the computer choice, otherwise the method returns false.

3. `getResultString()` method

The `getResultString()` method has two `int` parameters and returns a `String`. There are four possible `Strings` which can be returned by this methods:

```
final String PAPER_WINS = "paper covers rock";
final String ROCK_WINS = "rock smashes scissors";
final String SCISSORS_WINS = "scissors cut paper";
final String TIE = " you chose the same as the computer";
```

The `String` which is returned depends on the values of the two `int` parameters. If the two parameter values are equal then the method returns TIE.

4. `start()` method

The `start()` method calls the previous methods for running the application. You will also need to implement the following to complete the application:

- Printing the prompt and reading the input from the user for the name
- Printing the prompt and reading the input from the user for the choice of rock, scissors, paper, or quit
- While the user is playing, display the player choice, determine who wins, and display the results
- Printing a message to display the goodbye message when the user chooses to quit the application

The skeleton code is found in:
`ictgradschool.industry.controlflow.rockpaperscissors.RockPaperScissors.java`