PGCert IT: Programming for Industry

Arrays and Classes

Exercise One: Arrays

Do the following **on paper**!
Consider the following array:

```
int[] numbers = {-9, 2, 7, 5, 124, -5, 1, 144};
```

1. What would be the output of the following statements?
   a. `System.out.println(numbers[0]);`
   b. `System.out.println(numbers[numbers.length - 1]);`
   c. `System.out.println(numbers[numbers[1]]);`
   d. `System.out.println(numbers[0] * numbers[1]);`
   e. `System.out.println(numbers.length);`

2. Declare an array of **`double`**s named `amounts`.

3. Construct the `amounts` array declared in 2) above, big enough to hold 100 elements.

4. Write a Java statement which assigns 22.75 to element 0 of the `amounts` array.

Exercise Two: Looping through an array

Do the following **on paper**!
Complete the method below so that it adds up all the elements in the **`values`** array, then returns the total value. You will need to fill in the gaps.

```
private double getTotal(double[] values) {
  // You complete this for loop
  double totalValue = 0;
  for (_____________;_____________________________;_____) {

          ____________________________________________________;
  }
  return totalValue;
}
```

Exercise Three: The deodorant class

Below is the definition of the `Deodorant` class. The skeleton code is found in: `ictgradschool.industry.arrays.deodorant.Deodorant.java`

```
public class Deodorant {

	private String brand;
	private String fragrance;
	private boolean rollOn;
	private double price;

	public Deodorant(String brand, String fragrance, boolean rollOn,
		double price) {

		this.brand = brand;
		this.fragrance = fragrance;
		this.rollOn = rollOn;
		this.price = price;
	}

	public String toString() {
		String info = brand + " " + fragrance;
		if (rollOn) {
			info = info + " Roll-On";
		} else {
			info = info + " Spray";
```

```
            }
            info += " Deodorant, \n$" + price;
            return info;
        }
}
```

1. In the `Deodorant` class, complete the method definitions for
    - the accessor methods `getPrice()`, `getBrand()`, `isRollOn()` and `getFragrance()`
    - the mutator methods `setPrice()`, `setBrand()`, `setFragrance()`
    - the boolean method `isMoreExpensiveThan(Deodorant other)`

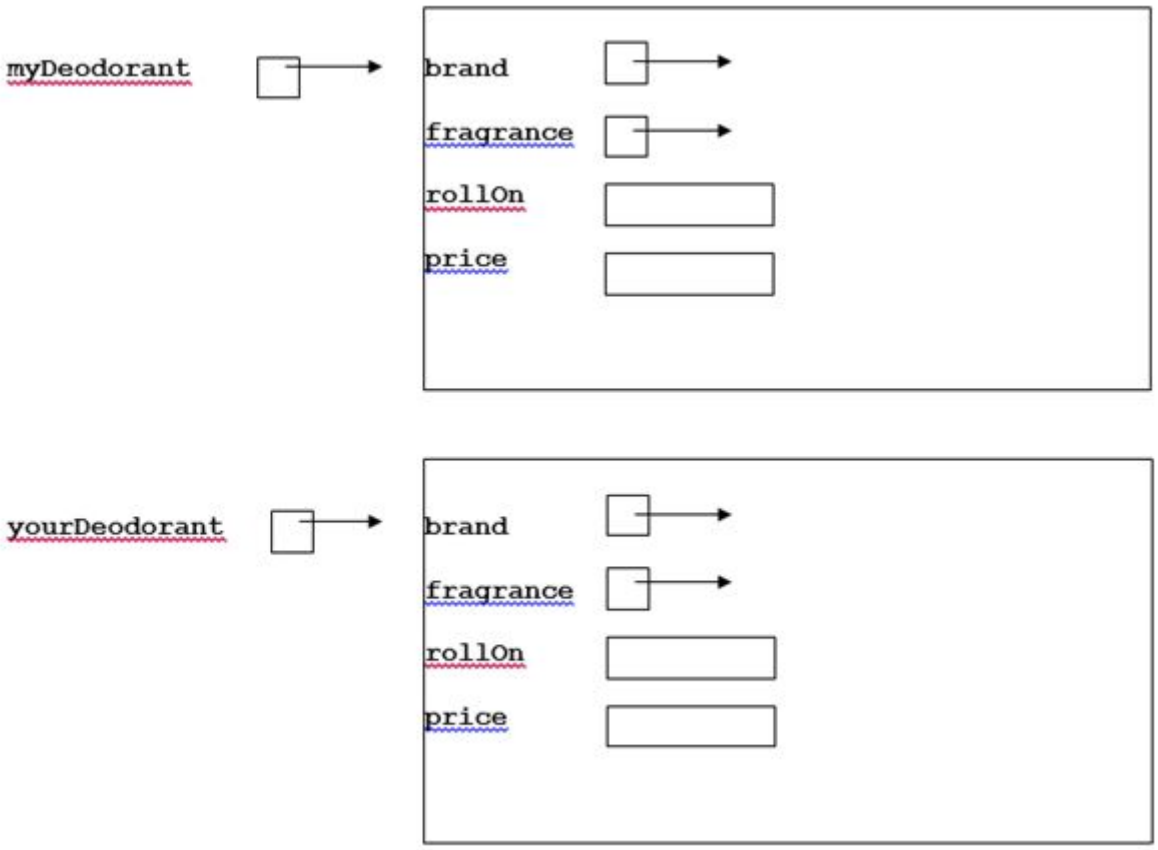
    Note: You can test your implementations using the supplied test class, `ictgradschool.industry.arrays.deodorant.TestDeodorant`.

2. Two objects of type `Deodorant` are created as follows:

    ```
    Deodorant myDeodorant = new Deodorant("Gentle", "Baby Powder", true, 4.99);

    Deodorant yourDeodorant = new Deodorant("Spring", "Blossom", false, 3.99);
    ```

    Complete the diagram on the next page illustrating the values that are stored in the instance variables for each of these objects. You should write **very clearly** on the diagram, as you are required to change the diagram when you complete part (3) of this question.



3. Given the two Deodorant objects created and initialised as in the diagram above, give the output when the following statements are executed.

   Note: You MUST also mark any changes to the instance variables clearly on the diagram above.

```
System.out.println("1. " + myDeodorant.toString());

myDeodorant.setBrand("Sweet");

yourDeodorant.setPrice(5.29);

System.out.println("2. " + yourDeodorant.toString());

if (myDeodorant.isRollOn()) {
      System.out.println("3. Roll On");
} else {
      System.out.println("3. Spray");
}

System.out.println("4. " + myDeodorant.toString());
```

```
if (yourDeodorant.isMoreExpensiveThan(myDeodorant)) {

      System.out.println("5. Most expensive is " +
          yourDeodorant.getBrand());
} else {

      System.out.println("5. Most expensive is " +
          myDeodorant.getBrand());
}
```

Exercise Four: The pattern class

The skeleton code is found in the `ictgradschool.industry.arrays.printpattern` package.

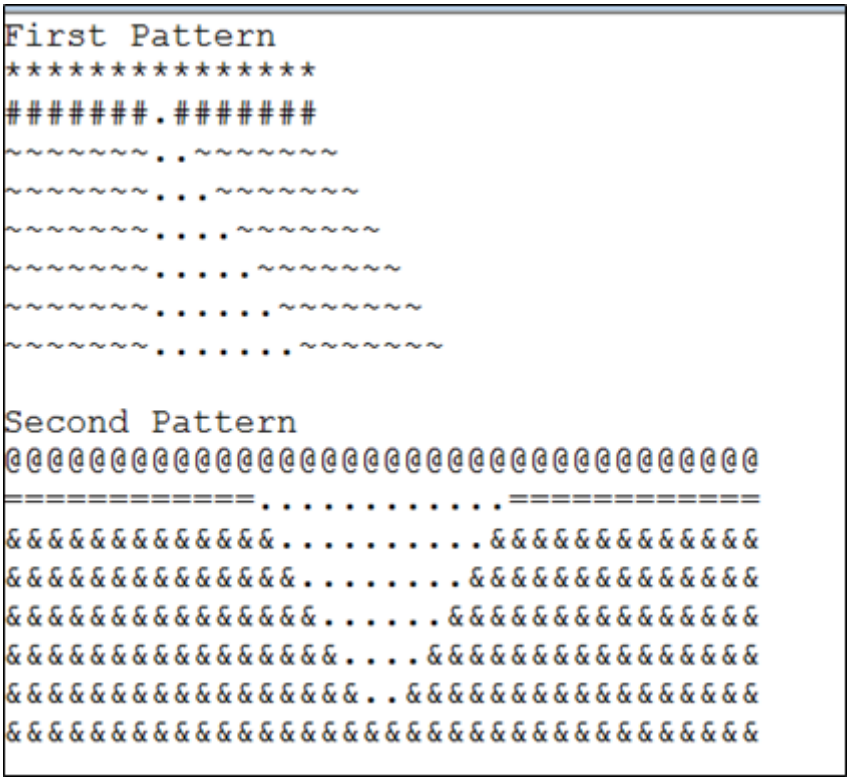
Open the file `PrintPatternProgram.java`. This class creates an instance of the `Pattern` class and calls the methods in the `Pattern` class to print different patterns.

The `Pattern` class defines a pattern. It consists of 2 instance variables: the pattern symbol and the number of repetitions of the symbol. Create the `Pattern.java` file and complete the class so that `PrintPatternProgram` can print the first pattern in the screenshot below.

Hint #1: You can create a new class in IntelliJ by right-clicking the package, and choosing New Java Class. Name it `Pattern`.

Hint #2: Look at the code that's commented out in `PrintPatternProgram` to see what methods your `Pattern` class needs to implement.

By calling the methods in the `Pattern.java` file, complete the `printPatternTwo()` method in `PrintPatternProgram` so that the second pattern is also printed, as in the screenshot below. This method must create `Pattern` objects in a similar way to the `printPatternOne()` method.



Exercise Five: The medals class

The skeleton code is found in the `ictgradschool.industry.arrays.medals` package.

Complete the methods in the `Medals.java` file so that when the `DisplayMedalResults` is run it produces the following output:

```
New Zealand has 4 gold medals, 12 silver medals, 13 bronze medals
Australia has 3 gold medals, 20 silver medals, 10 bronze medals
Canada has 6 gold medals, 5 silver medals, 7 bronze medals

Canada has the most gold medals
Australia is the overall winner
```

You need to write the following methods:

- The constructor
- The `toString()` method
- The `getCountryName()` method
- The `setCountryName()` method
- The `hasMoreGoldMedalsThan()` method

Exercise Six: The mobile phone class

The skeleton code is found in the `ictgradschool.industry.arrays.mobilephones` package.

Complete the `MobilePhone` class, and uncomment the marked lines in `DisplayMobilePrices.java` so that when `DisplayMobilePrices` is run, it produces the following output:

```
Jonathan has an Apple iPhone 4 which cost $899.95
Ann has an LG Optimus-P970 which cost $699.95
Adriana has a Nokia N97 which cost $599.55
Alastair has now purchased a new Apple iPhone 4 for $899.95
Alastair has the same type as Jonathan
Adriana wants a new phone
```

You need to declare the 3 instance variables and write the following methods:

- The `MobilePhone()` constructor
- The `getPrice()` and `setPrice()` methods
- The `toString()` method
- The `getModel()` and `setModel()` methods
- The `equals()` method
- The `isCheaperThan()` method

Exercise Seven: The lecturer class

The skeleton code is found in the `ictgradschool.industry.arrays.lecturers` package.

Complete the `Lecturer` class and uncomment marked lines in `LecturerProgram.java` so that when `LecturerProgram` is run, it produces the output as per the screenshot below.

You need to write the following methods:

- The `Lecturer()` constructor
- The `getName()` and `setName()` methods
- The `getStaffId()` and `setStaffId()` methods
- The `getPapers()` and `setPapers()` methods
- The `isOnLeave()` and `setOnLeave()` methods
- The `teachesMorePapersThan()` method
- The `toString()` method

Note:

1) You will need to check the `printLecturers()` method in the `LecturerProgram` class to see what needs to be done in your `toString()` method.

2) The instance variable `papers.length` will give you the number of papers that the lecturer takes.

```
Current Lecturers
-----------------
1.  id:86302 Sad Sack is teaching 2 papers.
2.  id:49123 Ali Katt is teaching 2 papers.
3.  id:40879 Earl Lee Riser is teaching 3 papers.
4.  id:50876 Candy Kane is teaching 4 papers.
5.  id:30869 Tom Katt is teaching 0 papers.
6.  id:30987 Carrie Oakey is teaching 2 papers.

Lecturers Currently on Leave
----------------------------
Earl Lee Riser
Carrie Oakey

Updated details for changed lecturer number 2
---------------------------------------------
Name: Crystal Ball
Id: 23456
Papers: CompSci101 CompSci105
Currently on leave

Most papers
-----------
Candy Kane teaches more papers than any other lecturer.
```

Exercise Eight: Movies

The skeleton code is found in the `ictgradschool.industry.arrays.movies` package.

Complete the code in `MovieProgram.java` as in Steps 1 - 5 below, so that it produces the following output when you run the code.

```
Movie Collection
================
Meet the Parents (2000), 107 minutes, Director: Jay Roach
The Parent Trap (1961), 129 minutes, Director: David Swift
Alice In Wonderland (2009), 109 minutes, Director: Tim Burton
Dark Shadows (2012), 113 minutes, Director: Tim Burton
The Iron Lady (2011), 105 minutes, Director: Phylliday Lloyd
The Help (2011), 137 minutes, Director: Tate Taylor
From Russia With Love (1963), 110 minutes, Director: Terence Young
The King's Speech (2011), 118 minutes, Director: Tom Hooper
Charlie and the Chocolate Factory (2005), 115 minutes, Director: Tim Burton
Easy Rider (1969), 94 minutes, Director: Dennis Hopper
Walk the Line (2005), 136 minutes, Director: James Mangold
Kaikohe Demolition (2004), 52 minutes, Director: Florian Habicht
Brokeback Mountain (2005), 134 minutes, Director: Ang Lee
Gladiator (2000), 154 minutes, Director: Ridley Scott
The Long Voyage Home (1940), 105 minutes, Director: John Ford
Happy-Go-Lucky (2008), 118 minutes, Director: Mike Leigh
The Big Wedding (2013), 89 minutes, Director: Justin Zackham
The Intouchables (2011), 112 minutes, Director: Olivier Nakache and Eric Toledano
Searching for Sugar Man (2012), 86 minutes, Director: Malik Bendjelloul

The most recent movie is: The Big Wedding (2013), 89 minutes, Director: Justin Zackham
The longest movie is: Gladiator (2000), 154 minutes, Director: Ridley Scott

Searching for Sugar Man was directed by Malik Bendjelloul.
Liberal Arts is not in the collection.
The Intouchables was directed by Olivier Nakache and Eric Toledano.
```

1. Declare and construct an array of 19 `Movie` objects.
2. Write the `printMoviesArray()` method. This method takes an array of `Movie` objects as a parameter and prints all the elements as per the screenshot above. Note that the `toString()` method in the `Movie` class can be called to obtain a `String` containing the instance variables of a particular `Movie`, formatted in the required manner.
3. Write the `getMostRecentMovie()` method. This method takes an array of `Movie` objects as a parameter and returns a reference to the most recent `Movie`. Note that the `isMoreRecentThan()` method in the `Movie` class can be used to determine if a `Movie` is more recent than another `Movie`.
4. Write the `getLongestMovie()` method. This method takes an array of `Movie` objects as a parameter and returns a reference to the longest `Movie`. Note that the `isLongerThan()` method in the `Movie` class can be used to determine if a `Movie` is longer than another `Movie`.
5. Write the `printDirector()` method. This method takes 2 parameters: the name of a movie, and an array of `Movie` objects. The method should loop through the array searching for the movie with the name that has been passed in as a parameter. If it finds a movie with that name, it should print out the director of the movie as per the screenshot above. If it cannot find a movie with the name that has been passed in as a parameter, then it should print out “not in the collection” as per the screenshot above.