# Beijing-Dublin International College

---

**AUTUMN TRIMESTER FINAL EXAMINATION - (2021/2022)**

---

**School of Computer Science**

# COMP2011J Programming Exam

**Dr. Rosemary Monahan**
**Assoc. Prof. Chris Bleakley**
**Dr. Seán Russell***

## Time Allowed: 90 minutes

### Instructions for Candidates:

Answer all questions.

**BJUT Student ID:**_____ **UCD Student ID:**_____

I have read and clearly understand the Examination Rules of both Beijing University of Technology and University College Dublin. I am aware of the Punishment for Violating the Rules of Beijing University of Technology and/or University College Dublin. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I accept the punishment thereof.

**Honesty Pledge:**_____ **(Signature)**

### Instructions for Invigilators

N/A

## Exam Project

To complete this exam you should download the Exam Project zip file from moodle. This file contains a IntelliJ IDEA project with some code for you to use and some input data.

## Project Contents

The important files and folders are listed here:
**Exam** - project folder

- **src** - Folder where you should create your classes
  - **Student.java** - This class is for you to use in question 3 of the exam, do not change.
- **test** - Folder where the test classes are stored. You can use these test cases to judge if you have completed some of the tasks.
  - Q1Test.java - A number of test cases assessing Question 1 a
  - Q2ATest.java - A number of test cases assessing Question 2 a
  - Q2BTest.java - A number of test cases assessing Question 2 b
  - Q2CTest.java - A number of test cases assessing Question 2 c
  - Q2DTest.java - A number of test cases assessing Question 2 d
  - Q3Test.java - A number of test cases assessing Question 3 a
- **names.csv** - A file containing some input data to be used in the tests

`names.csv`

This file contains 99 lines of text. Each line represents information about a single student and is separated by commas. Each line contains the following information in this order:

- The students given (first) name
- The students family name
- The students date of birth (in the format YYYY-MM-DD)

## Tests

The test cases in the files above do not cover all of the situations that your code will be tested for. If you pass all of these tests, then you are guaranteed that you have **passed the exam**. To ensure that you have earned a good grade you need to carefully consider the contents of each question and make sure that your code satisfies all of the requirements.

## Rules

- This exam is open book, this means that you are allowed to use the course notes and any textbook during the exam.
- This is also an **individual** exam. All submissions will be compared and any students found to have cheated will be reported to the School of Computer Science plagiarism committee and potentially to the university plagiarism committee.
- The consequences of plagiarism are a best case of **failing** the exam, potential **fines**, **not being allowed** to repeat the module for a year, or a worst case of being **removed from the degree**.
- You should only submit working code. If your code does not compile or work correctly you should remove the part that is not working. You will **fail** if you submit code that **does not compile**.
- The classes you define should have good encapsulation. A portion of your grade in **every** question is based on this.

## Submission

In order to complete the exam, you must submit your code to the exam quiz on the class page of csmoodle.ucd.ie. It is your responsibility to ensure that this is done correctly. The quiz close automatically at the end of the exam and will not be reopened for any reason. If you are having trouble submitting to moodle, you must email your code to be **before** the deadline has passed.

If your exam is not submitted correctly and on time you will **fail the exam**.

## The Exam Quiz

The exam is similar to the quizzes that you have completed in the past, however you will not receive any feedback or score before the quiz is completed. You must ensure that you have tested you code fully before submitting.

## Question 1: Basic Programming Problem (Class and Static Method)

a. Define a class called `Q1`, containing a class/static method called `boxes`. This method should return a `String` containing a pattern and take 2 `int` and 2 `char` parameters. The first `int` is the height of the pattern, the second `int` is the width, and the `char` parameters are the characters that should be used in the pattern.

The method should return a string containing representing three boxes shown using the first char parameter with the rest of the shape filled in with the second char parameter.

A runtime exception should be thrown in any of the following situations (The class `IllegalArgumentException` would be a good one to use, but you can use another or define your own if you want):

- The char parameters are equal
- The width or height are not in the range 7-20 (inclusive)

The following are examples of the string returned by the code when the method is called with the parameters shown.

```
Q1.boxes(7, 7, '+', 'o');
```

```
+++++++
+ooooo+
+o+++o+
+o+o+o+
+o+++o+
+ooooo+
+++++++
```

```
Q1.boxes(9, 9, '-', '|');
```

```
---------
-|||||||-
-|-----|-
-|-|||-|-
-|-|||-|-
-|-|||-|-
-|-----|-
-|||||||-
---------
```

When the size of the pattern is increased in either direction, the space in the center of the shape is expanded.

**(20%)**

**(Question Total 20%)**

## Question 2: Basic Programming Problems (Class and Instance Methods and Variables)

a. Define a class called `Q2`.

#### Instance Variables

The class should have the following two instance variables:

(a) an `ArrayList` with the type parameter `String` called `arrayList`

(b) a `double` array (not an `ArrayList`!) called `doubleArray`

Note: You may find it necessary to add other instance variables to the class in order to complete later functionality.

#### Constructor

The class should have a constructor that initialises both instance variables. The constructor must accept one input parameter, an `int` which defines the initial size of the `double` array. This parameter **must** be positive, and a suitable run time exception should be thrown if it is not.

#### Accessor Methods

The class should also contain accessor (getter) methods for the two instance variables described above.

#### Encapsulation

The class must be defined with consideration given to encapsulation (so consider the visibility of instance variables, methods and constructors and the mutability of instance variables with accessor methods.

**(10%)**

b. Continuing to work on the class `Q2`, define as many mutator methods with the name `add` as you need to be able to accept a single parameter of each of the following types:

- `int`
- `double`
- `float`
- `String`
- `Object`

The method does not need to return any value.

**Parameter Types**

If the value passed as a parameter is numeric data (i.e. `int`, `double`, and `float`), this should be inserted into the `double` array. If the value passed as a parameter is not numeric data (any other type of data), this should be added to the `ArrayList`. When an `Object` is received, call its `toString` method and add the `String` to the `ArrayList`.

**Numbers in Strings**

If the parameter is a `String` that contains a number, you do not need to convert this value. It should be stored in the `arrayList` as a `String`.

**(20%)**

c. Continuing to work on the class `Q2`, define as many methods with the name `contains` as you need for the types below. All of the methods should return `true` if the passed value is present in either the `ArrayList` or `double` array, and `false` if it not present in either. The methods should be able to accept parameters of the following types:

- `int`
- `double`
- `float`
- `String`
- `Object`

**(10%)**

d. Continuing to work on the class `Q2`, define a `toString` method. This method must override the `toString` method inherited from the Object class. This method should return a String containing all elements from the `ArrayList` followed by all elements from the `double` array as a comma separated list.

**Note**

Do not return empty values in the `double` array in the `toString` method.

**Example**

Code:

```
Q2 test = new Q2(2);
test.add("Java");
test.add("Exam");
test.add(1);
test.add(2);
String re = test.toString();
System.out.println(re);
```

Result:

```
Java,Exam,1.0,2.0
```

**(10%)**

**(Question Total 50%)**

## Question 3: Processing Files and Objects

a. Define a class called `Q3`.

### Data From a File

Define a class/static method named `readStudentsFile`. This methods should take a `String` as a parameter (containing a file name) and return an `ArrayList` of `Student` objects. The method should read all of the data from the file named in the parameter, use this data to create Student objects and add them to the array list in the same order as they appear in the file.

### Sorting Students by Age

Define a class/static method named `sortAge`. This method should take an `ArrayList` containing `Student` objects as a parameter and sort the student objects from oldest to youngest.

**Notes:**

- Date values represented as `Strings` in the format YYYY-MM-DD do not need to be converted into another data type to be sorted (they can be sorted as `Strings`)
- You cannot make any changes to the Student class (these changes will be ignored when your code is being tested)

### Sorting Students by Names

Define a class/static method named `sortFamilyGivenName`. This method should take an `ArrayList` containing `Student` objects as a parameter and sort the student objects alphabetically based first on their family name and then on their given name (only when family names are the same).

### Finding Students

Define the class/static methods named `getMedianAgeStudent`, `getYoungestStudent`, and `getOldestStudent`. The methods should all take an `ArrayList` containing `Student` objects as a parameter and return the `Student` object that is the median age, youngest or oldest respectively.

**(30%)**

**(Question Total 30%)**

**(Exam Total 100%)**