Sarajevo School of Science and Technology
CS/IS 160 Programming and Problem Solving II
June 2015

Final Exam Review

<u>Final exam information:</u>

- Scheduled for Friday, June 26
- Study material from textbook Chapters 1 – 11 and 16
- Format similar to our previous exams

The exam has 7 problems with the following format:

- Problem with short answers (study theory and questions at the end of chapters)
- Problem with simple declarations, statements, and code
- Three problems to implement static methods (for example, implement a function to process arrays, lists, matrices, strings, *etc.*)
- Two problems dealing with classes, inheritance, interface, etc. (OOP concepts)

Practice problems

<u>Product of diagonal entries of a square matrix.</u>

Implement a static method `diagProd`, which returns the product of the diagonal entries of a square matrix. For example, if the input matrix is given as

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

then the sum of its diagonal entries is $1 \times 5 \times 9 = 45$.

Use the following method prototype, where `matrix` is an input integer square matrix:

```
public static int diagProd( int[][] matrix )
```

<u>Filtered counter for a square matrix.</u>

Implement a static method `countGreaterThan`, which returns the number of elements of a square matrix that are greater than an input threshold value. For example, if the input matrix is given as

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

and the threshold value is 5, then the returned count is 4 (there are four numbers in the matrix that are greater than 5).

Use the following method prototype, where `matrix` is an input integer square matrix and `val` is the threshold value:

```
  public static int countGreaterThan( int[][] matrix, int val )
```

Downsample an array.

Downsampling by a factor of *n* is a digital signal processing (DSP) technique in which we take an input array and return a smaller array created by sampling every *n*th element, starting with the first one.

Implement a static method `downsample`, for which the inputs are an array and a downsample factor, while the output is the downsampled array. For example, if the input array is (1, 2, 1, 0, 5, 6, 7) and the factor is 3, then the output array is (1, 0, 7).

Use the following method prototype, where `array` is an input integer array and `n` is the downsample factor:

```
        public static int[] downsample( int[] array, int n )
```

Equal arrays.

Write a static method `equalArrays` which tests if two arrays of integers are identical, that is, if they contain the same elements in the same order. Use the following method prototype, where `a` is the first array and `b` is the second array:

```
       public static boolean equalArrays( int[] a, int[] b )
```

Equal lists.

Write a static method `equalLists` which tests if two lists of integers are identical, that is, if they contain the same elements in the same order. Use the following method prototype, where `a` is the first list and `b` is the second one:

```
public static boolean equalLists( List<Integer> a, List<Integer> b )
```

<u>String max run.</u>

Write a static method `maxRun`, which given an input string returns the length of the longest *run* in the string. A *run* is defined as a series of zero or more adjacent characters that are the same. So the max run of "xxyyyz" is 3, and the max run of "xyz" is 1. Note that the string instance method `charAt(int index)` can be used to obtain a character located at the index location in the string.

Use the following method prototype:

```
public static int maxRun( String str )
```

<u>Shuffle a string list of names.</u>

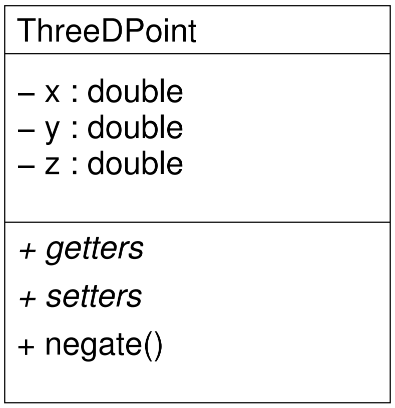
Write a static method `shuffleNames` that takes a list of String objects and rearranges them randomly inside the list. Use the following method prototype:

```
public static void shuffleNames( List<String> names )
```

<u>Point in 3D space class.</u>

Implement a simple `ThreeDPoint` class that models points in 3D vector space (each point has three coordinates `x`, `y`, and `z`, stored as private double variables).

The UML diagram for the class is shown below.



a) Implement a default constructor that sets coordinates to zeros.

b) Implement the getter and setter method for the `x` variable.

c) Implement the public method `negate`, which flips the sign of each vector coordinate.

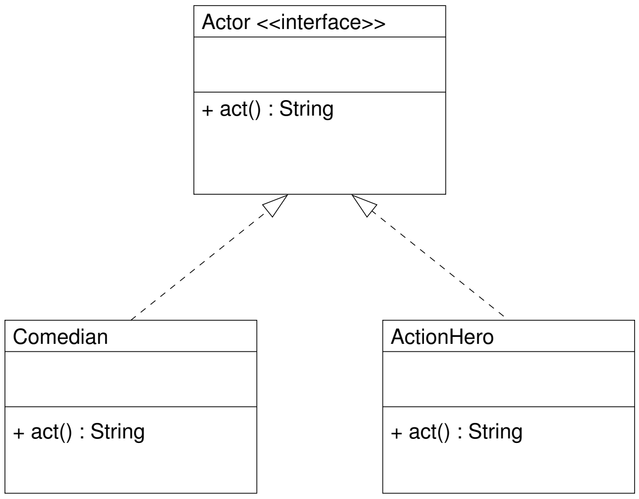
d) Implement the public instance method `toString` to return the following string representation of the `ThreeDPoint` object "(x, y, z)", where x, y, and z are the values of the vector coordinates.

Use the following instance method prototype:

```
public String toString()
```

Polymorphism with Actor classes.

We have the class hierarchy shown below and implemented as practice for Test 3. Here, the `Actor` is an interface and the `Comedian` and `ActionHero` classes implement the interface. The `act` method for the `Comedian` class should return the string "I'm Austin Powers," while the `ActionHero` class should return "My name is Bond, James Bond."



Write a main method that creates an `Actor[]` array named `myMovieCast`, with three elements and populate it with one comedian and two action heroes. Then iterate over each member of the array, and print out the string output of its act method call.

Now repeat the same for an array list of Actors `ArrayList<Actor>` named `myMovieCastList.`