# Homework 04

Version: 1.6

Due: 2016-10-27T23:59:59

# TODO

Download zip (hw4.zip)

# Introduction

Civilization (https://en.wikipedia.org/wiki/Civilization_(video_game)) is a turn-based strategy game centered around founding and building a civilization. More commonly referred to as Sid Meier's Civilization, the game has been around since 1991 and has since developed somewhat of a cult following. This semester we are going to be implementing our own version of this game for your homework assignments and by the end of the semester you will have a working game you can show off to your friends! If you are not familiar with Civilization, there is a free version (https://en.wikipedia.org/wiki/Freeciv) that you can play in order to get familiar with the game.

This assignment will get you working with Collections and Exceptions. This project will:

- Assess your understanding of collections on a conceptual level
- Test your ability to implement a custom collection in a Java program
- Require proper usage of java Exceptions and Exception-handling

# Problem Description

You've now had some practice writing java Classes that use inheritance and polymorphism in a java program. It's time to get a little more practical with our inheritance and polymorphism! We'll do that by working with java Collections and java Exceptions.

# Before we begin…

- Like all homework descriptions, it is very important you read and understand this entire document.
- It is important to note that you will not need to create a new repository for this homework. You will be working in the same git repository you set up for the first homework.
- This homework will be very focused on details. So make sure you read the description carefully and implement every detail described!
- Note that many of the instance variables will require getter and setter methods (since all of your instance data should be private) that we did not explicitly describe in the homework description. If you complete all of the classes correctly then `CivilizationGame.java` should compile and run no problem! If you receive errors about missing methods, you should add that functionality to your class.
- Most of this code is revised from HW03 code, so you will need to replace any duplicated files in your repository with the new versions.

# A little about Collections

"Collections" are 2 things:

- First: Collections are a group of conceptual structures and ideas that are used in discrete math/computer science theory. They have to do with the storage of data. They are lists, sets, maps, trees, and much more! These concepts are very commonly implemented in programs since they are such useful ideas for storing and manipulating information, which brings us to the second point…
- Second: The Collections Framework is a group of java standard library classes and interfaces. The interfaces define the requirements to implement the abstract data types of collections. The classes implement the abstract data types of collections. There is a List interface, a Set interface, etc. Each may have multiple classes implementing the interfaces. One of these you may already be familiar with since it has been mentioned in class before: ArrayList (it is a List that is backed by an array).

In this assignment, you will in essence be implementing the Set class. A Set is a data structure that holds zero to infinity entries, holds only unique entries, and those entries have no ordering.

# Compiling and Running

- **This project will not compile in its entirety right off the bat**
- As you work, you will need to individually compile the files you write to make sure that they work! So remember to compile often so that you don't have to deal with a mountain of errors! Keep in mind that when you compile only one file, it does not compile some of the other ones that it references. So, you will probably get alot of cannot find symbol errors. Don't worry about them, they appear because some dependency classes aren't compiled. You may also get warnings that methods aren't being overridden or something if you use the @Overide tag. Once again, that is because the superclass didn't get compiled if you compiled that file only.
- This project makes use of several different packages to manage its many files. As a result, compiling and running this project is a bit different from projects you have worked on in the past.
- Since our files are in many different packages, we have to make sure they get linked together properly. We do this by setting the class path when we compile and when we run.
- To compile the program in its totality run this command if using bash:

```
$ javac -cp src/main/java src/main/java/*/*.java
```

- AND IF THAT ONE DOESN'T WORK, this one should work for sure.

```
$ javac -cp src/main/java src/main/java/runner/*.java src/main/java/model/*.java src/main/java/view/*.java src/main/java/controller/*.java
```

- That command instructs java to compile all of our files while considering src/main/java to be our classpath.
- To run the program, you should run:

```
$ java -cp src/main/java runner.CivilizationGame
```

# Checkstyle

- Remember that you may lose checkstyle points for this homework! So, make sure that you are frequently running checkstyle on your code to ensure that you don't lost unnecessary points.
- There are currently two checkstyle errors in the homework that you are not responsible for. These are caused by some provided constructors taking in more than 7 parameters. You will not be penalized for these two checkstyle errors.
- When writing MilitaryUnit.java, the constructor will also take in more than 7 parameters. You will not be penalized for this third checkstyle error either.
- **The checkstyle cap on this assignment is 50 points**, where you lose one point per error.

# Solution Description

- Note: When making new classes, remember that we are working with multiple packages. You will have to specify what package each file belongs to at the top of the file. Read through some of the provided files to see how to do this. It is also suggested that you look through the files to understand the interplay between all of the different parts.
- Note: You might notice that all of the classes in the model package, with the exception of Model, do not have an access modifier, making them package private. This is by design to prevent access to these classes from outside the package. Any class you create for this project should have package private access.

## MySet.java

Your first custom java collection! This class will need to implement the provided SimpleSet interface and should be backed by an array. As you may have inferred, MySet is going to act like the abstract data type of a “set”. This means that it is going to not have any particular ordering, and it is only going to contain unique elements. Look at the javadocs for each method in the SimpleSet interface for details on how to implement this class!

NOTES:

- Since your MySet class will be backed by an array, the entries will “have an ordering” in the array; however, as long as the programs using your class treat your class as though the elements have no ordering then your class meets the blueprint provided by the abstract data type of a Set.
- MySet should be able to hold an any number of elements.
- MySet uses generics. In order to create an array of the generic type you must first create an array of Objects and cast it to an array of the generic type because java will not allow you to directly create an array of the generic type.
- MySet will throw an ElementDoesNotExistException in a number of places. This is a custom exception class in the model package. It is recommended that you look at it to get a full understanding of what your code will be doing.

## FishingShack.java

If you recall, from HW03 the FishingShack class tracked several fish using an array. Well, you guys are clearly too advanced to be storing fish in arrays; so, we are going to store the Fish in our new MySet class! Implement the getFish() and replenishFish() methods in FishingShack.java. See the javadocs for those methods for implementation details.

- Note: In this case, you will need to add a random Fish to a MySet. Even if that Fish is equal to a Fish that’s already in the MySet you still need to add a Fish to the MySet!

## Technology.java

Our Civilizations should also know what Technologies it has researched. We will call these “skills”. In the Technology class you are going to write the gainATech() method. See the javadocs for details.

- Note: In this case, you will need to add a random Skill to a MySet. Even if that Skill is equal to a Skill that’s already in the MySet you still need to add a Skill to the MySet!

# Conclusion

- If everything worked out right, you should now be able to compile and play your game the same as homework 3! The only difference from homework 3 is that there should be a list of “skills” that is displayed along with the rest of your Civilization’s information. We’ll be iterating on this in future homeworks, so make sure you become familiar with how it works, and all of the different moving parts.

# Verifying Your Submission

Please be sure that any code you push compiles and runs through the command line! Pull from your repository and make sure everything is working how you want it!