Homework 01

Due: 2016-09-15T23:59:59

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 ramped up on Java syntax and the basics of the language. This project will:

- Test your ability to use and manipulate variables and values in the Java language
- Test your ability to solve problems by utilizing control structures
- Test your ability to use basic I/O through the console
- Test your ability to create and use arrays

Problem Description

Civilization is a turn-based strategy game revolving around moving units around a map to conquer other civilizations to reach the highest point in the technology tree (http://civilization.wikia.com/wiki/List_of_technologies). For this first assignment we will focus on creating a small command line version of the game that interacts with a player. - Note: This covers a simple one player version of this game. You will not be implementing other players or AIs that can affect the player’s civilization.

Before we begin..

We will be using git for homework submissions. Git is a Version Control System. I highly suggest you learn how to use git through either here (https://www.codecademy.com/learn/learn-git), here (https://githowto.com), or by doing your own research and reading. We outline step by step how to get everything set up below for you.

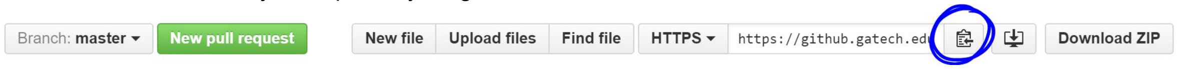
Do not change your username on github. It should remain your GT username as was given to you by the institute. For example, I changed my email a long tim ago to be tjhartman@gatech.edu, but my github account name is still thartman3. If you change your name you will just be making everyone’s lives more difficult this semester. So just don’t do it.

Setting up a Git Repository

Before you start coding, you need to do a little bit of set up in order to be able to turn your homework in through git.

In your web browser…

1. First and foremost, if you do not have git installed, you need to install it (https://git-scm.com/).
2. Once you have it installed, go to github.gatech.edu and login with your GT credentials.
3. When you reach the landing page, on the right side there should be a “Your Repositories” list. Click on the button that says “New Repository”

4. In the textbox for Repository Name put “civilization_fall16”. For example, to access my repository my url would be github.gatech.edu/thartman3/civilization_fall16.
5. Next, make your repository “private.” ***This is very important!*** If your repository is public other people will be able to see your code. This will be considered an honor code violation.
6. Next, check the box next to where it says “initialize with a README”
7. Next, in the drop down menu where it says “Add .gitignore:” select Java from that dropdown menu.
8. Now select “create repository”
9. You should have been taken to the location of your repository on github. You will see a bar that looks like the bar shown below. Click on the button circled below. This copies a link to your clipboard. 

10. Go to “Settings.” This is your repository’s settings. You could change stuff here if this were a repository for personal use, but since it’s for homework leave everything unless we tell you to change it!
11. On the left hand side select “Collaborators”. You will be asked to confirm your credentials.
12. Add me, “thartman3”, as a collaborator. I will add other TAs as collaborators for you. **There should not be any other collaborators on your homework repository. If there are others this will be considered an honor code violation.**

Now, inside of your command line…

1. Open up the command line and `cd` into a directory where you want to keep your homework.
2. Type `git clone` and then paste the link you copied from github in step 9 above.
   - For example, I would type `git clone https://github.gatech.edu/thartman3/civilization_fall16.git`
   - It might ask you to type in your credentials again. When you start to type your password it won’t show up on the screen, but just type it out correctly and hit enter.
3. `ls` or `dir` and you should see a folder called `civilization_fall16` in your directory!

Great! Now you have basic set up done.

Your first push

In order to start writing code, you actually need a file to start with. Download Civilization.java (./Civilization.java) into your civilization_fall16 directory.

Now do the following:

1. `cd` into civilization_fall16 in your command line.
2. Type `git status`
   - You should see that you have an “Untracked File” called Civilization.java
   - We want to add this file to be “staged for commit.” We’ll get into what that is in just a bit.
3. Type `git add Civilization.java`
4. Type `git status`
   - Now you can see that Civilization.java is listed as a “change to be committed”
5. Now we want to commit our change. What this means is that on our local machine (our laptop, for example) we changed our project and we want to make sure that change gets reflected in our repository on github!
   - To commmit type `git commit -m "My first commit and adding Civilization.java"` . The part in quotes is your commit message. This message will tell anyone who takes a look at your repository’s commit history what you changed or did every time you committed.
   - Now that we have committed our changes we actually want to “push” them up to our repository on github.
6. Type `git pull origin master` . This brings your local version of your repository on your machine up to date with the remote version of your repository on github. Alwasy pull before you push!

7. Type `git push origin master` . This pushes all of the changes you committed on your local repository up to the remote repository.

If you go back to github.gatech.edu and go look at your repository you will see that Civilization.java is now in your remote repository!

As you work…

Any time you get some code written and running you should push your changes up to the remote repository:

1. Type `git status` to see what changes are staged or unstaged
2. Type `git add *` to add all changes you have made to be staged for commit
3. Type `git commit -m "My message"` to commit your changes to your local repository.
4. Type `git pull origin master` to make sure you are up-to-date with the remote version of the repository.
5. Type `git push origin master` to push your changes up to the remote repository.

Solution Description

Starting the Game

- When the game is first started, you should ask what Civilization the user wants to lead. Based on the Civilization the user selected, they will be referred to as that Civilization’s leader throughout the game. Users can choose from the following list of Civilizations with the leaders’ names in parentheses: American (George Washington), Zulu (Shaka), English (Queen Elizabeth I), or Chinese (Wu Zetian). Be sure to keep track of which civilization the user selects!
- Each civilization has the following properties that must be updated and changed as gameplay continues:
  - **Array of cities**- You need to keep track of the cities the player settles in an array.
    - The user starts with 1 city. Be sure to ask the user what they want to name their city.
  - **Number of Attacks** - You will need to keep track of how many times the player attacks another city
    - The user starts with 0 attacks.
    - It is not possible to have fractional attacks.
  - **Gold** - Money! Used to buy things. It is possible to have fractional portions of money (as in I can have 0.5 gold). Be sure to incorporate this into your solution!
    - The user starts with 20.5 gold
  - **Resources** - Used to build or maintain things. It is possible to have fractional portions of resources (as in I can have 0.5 of a resource). Be sure to incorporate this into your solution!
    - The user starts with 30 resources
  - **Happiness** - The happiness level of the people living in the player’s Civilization. Influenced by resources available and military activity.
    - The user starts with 10 happiness
    - There is no such thing as 1/2 happiness.
  - **Military Units** - Troops that attack other cities. Once a troop is used they need to rest up at home, so each military unit can only be used once.
    - The user starts with 0 military units.
    - Military units are whole units, we never make just half of a unit.
  - **Technology Points** - Each civilization wants to be on the cutting edge. Gain technology points to develop more advanced technologies than other civilizations.
    - Technology points are going to be whole numbers.

Gameplay

- Each turn you should give an update to the user about each of their properties. Be sure to format this nicely. If something is a decimal, keep the number of decimal places to 2.
- Each turn the user receives:
    - The user always receives 1 resource. If the population's happiness is above 20, then the user receives 5 resources for each city they have.
    - 3 gold for each city they have
    - +1 happiness for their population if the number of resources they have are divisible by 2.
    - -3 happiness for their population if the number of resources they have are not divisible by 2.
- The user then has the option to perform one of the following actions:
    - ***Settle a City***- Allows the user to establish a new city. Print out a list of cities the player already has. The player must name their city. A player can have a maximum of five cities and a minimum of one city.
    - It costs 15.5 gold to build a city.
    - ***Demolish a City***- Allows the user to demolish a city. Remember a player must retain at least one city. Print out a list of the cities that can be demolished and allow a user to select which city they want to be demolished.
    - When a city is demolished the user gains 1.5 resources.
    - When a city is demolished, there should not be any `null` s or empty strings printed out when a list of the cities are printed out later.
    - ***Build Militia***- The user spends 5 gold and 3 resources to increase the number of military units the user has by 1.
    - ***Research Technology***- The user spends 50 gold and 2 resources to receive +1 Technology points.
    - ***Attack Enemy City***- The user attacks an enemy city and loses 6 military units and 3 happiness. Happiness can go negative. The user then gains 10 gold from the city it attacked.
    - ***End Turn*** - The user can choose to just skip any actions this turn and directly end the turn.
- If the user does not have enough gold, resources, or military units to perform an action, print out that they cannot perform that action and their turn is over. The only exception to this is happiness, happiness can go negative and is not spent like gold or resources.

Ending the Game

- The player wins once they reach 20 technology points or have attacked an enemy city 10 times.

Tips, Tricks, and Warnings

- You are NOT allowed to use the `Arrays` class nor the `ArrayList` class. You are never allowed to use anything that trivializes the assignment.
- If your code does not compile it is an automatic 0.
- If your code crashes during normal game play it will be a very heavy deduction in points.
- You can assume that during gameplay if you are expecting a number as input you will receive a number as input. You are not expected to handle mismatch types as input.
- If you find your code is getting long and hard to follow, try breaking up chunks of your code into methods to make it easier to follow! This is in no way a requirement but is something that is encouraged and will make your life a little easier.
- Remember to frequently compile as you go. Do not write your entire assignment and then check to see if it works! Work smarter, not harder.
- Run checkstyle! This homework it does not affect you, but in future homeworks style errors will be a reduction in points. One point per style error will quickly add up.
- Start early. Ask for help early. Note that on the day homeworks are due the TA lab traditionally gets very busy!

Checkstyle

Peruse the CS1331 style guide here (http://cs1331.gatech.edu/cs1331-style-guide.html). This also has the instructions for how to run checkstyle on your code. Here's a more step-by-step rundown:

- Go to the style guide (http://cs1331.gatech.edu/cs1331-style-guide.html)
- Right click on the link that says `checkstyle-6.2.2.jar` and select “save as” or “save link as” or whatever it is your browser suggests.
- Save the jar file in an easily accessible location.
- Copy the jar file into the same directory as the `.java` classes you want to check.
- Open the command line and `cd` into the directory containing your `.java` files and the checkstyle jar file.
- Type and run `java -jar checkstyle-6.2.2.jar MyJavaFile.java` replacing `MyJavaFile.java` with the name of your java file.

What gets printed out is the list of all style errors your code has. Do your best to fix all of them and start coding in the correct style. Each individual checkstyle error results in -1 points on your assignment.

Maximum number of points you can lose on Checkstyle this assignment: **0**

Turn-in Procedure

In order to submit your assignment you need to ensure that your working code is pushed to your remote repository by the due date! Follow the instructions outlined above to push your code to your remote repository!

You should only have one `.java` file: `Civilization.java` .

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!