Option 1. Greenfoot Project: "Amazing Game!"

Create a Greenfoot Scenario. This is your chance to show off your creativity and your unbelievable programming skills.

You will create a game (or simulation) from scratch using Greenfoot. What kind of game will you create? That's up to you. Please don't use the excuse, "I can't think of anything." You have played so many video games you are an expert!

Be as creative as you want. You may look at http://www.greenfoot.org/ to find ideas for scenarios, but of course you cannot copy the game completely. If you want to recreate a commercial video game that is fine, but you must check with the teacher first. Don't attempt something that must be complicated in order to run at all.

All complex programs that run begin from a simple program that runs!

No nudity, killing of beings, or profanity.

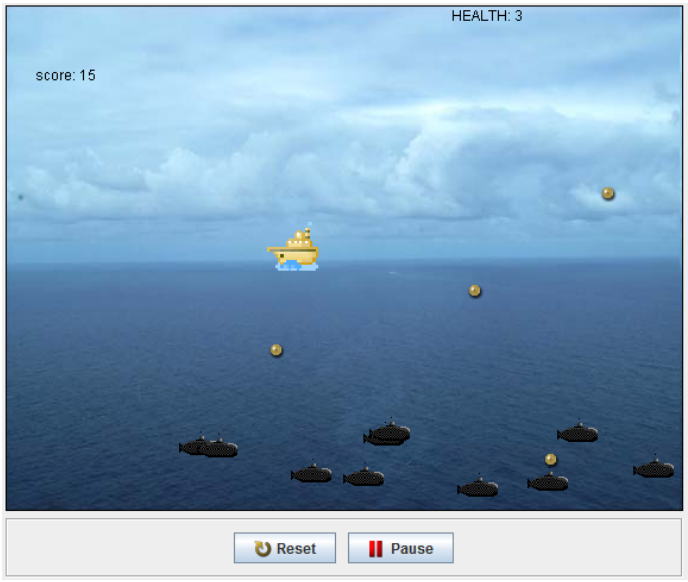
Before you Begin: First, before you begin programming, you have to submit a paragraph with your idea before starting.

Time is of the essence. Creating a game can take awhile, especially the troubleshooting stage when you have to fix errors in your code.

Check with the teacher for the due date.

Program Requirements: Look at the Rubric. Look at the Rubric. Again, look at the Rubric so you can get full credit.

Saving And Sharing Your File: When you are done, give a compressed (zipped) copy of your folder to the teacher via flash drive. Put the scenario on the Greenfoot site, http://www.greenfoot.org.



This is an example of a game. Use the left and right arrows and the spacebar.
Want to play it? Go to: http://greenfootgallery.org/scenarios/667

Help: Useful classes, including Mover, Counter and AnimatedActor can be found here:
http://www.greenfoot.org/doc/support_classes

Name: _____

End-Semester Final Greenfoot Project Rubric	Max	Points
Required Greenfoot elements.	30	
Actor and World Images There are at least TWO Actor classes. The background fits with your game.		
Moving The Actor: movement() method The protagonist class of the video game uses a method called movement() that is called in the act() method. This method contains the information for pressing keys, etc.		
Initializing Objects Upon Startup The World subclass uses its constructor to create at least one object in the Greenfoot scenario upon initialization.		
Disappearing Objects One of the actor objects in the world must disappear if it touches the edge of the world or touches another object.		
Keeping Score Your scenario must have a scoring system.		
Game Over – Stopping the Scenario The scenario must stop if a player loses or gets hit a certain number of times.		
Choose at least 2 of the following Greenfoot elements. Additional ones may be eligible for extra credit.	10	
Checking for Collisions Use a method called checkCollisions() somewhere in your Greenfoot scenario that checks to see if two objects are intersecting.		
Displaying Score Your Greenfoot scenario displays the score to the user. Check out the Counter class.		
Precise Movement More precise movement has been programmed into the game by flipping or rotating the actor to face the direction of movement.		
More keys Added keys beyond left and right.		
Required Java Elements.	10	
Two types of loops (for, while, do, 'for each')		
Parameters		
Methods, return types		
Fields (instance variables)		
Constructors that do something		
Choose at least 5 of the following Java elements:	10	
array, List or ArrayList; Input from file; Output to file; this; Explaining well how borrowed code works; Applet (.jar); Using Math library methods with reason; Sound; Collections; Casting		
Overall ratings.		
Game Playability and Goal The video game works, must also be playable, and easy to play. The game has a point and there is a clear goal.	5	
Code Quality The class and method structure is organized and logical. Variable names make sense.	5	
Comments and Java Doc Comments The programmer used appropriate comments and Java Doc comments in the code.	5	
Development Process Key qualities: Planning (including the initial paragraph with the ideas), effort, independence, tireless bug finding, challenging yourself at your level.	10	
Difficulty and Complexity Does it need many Java or Greenfoot elements? Does it have complexity including many objects and methods? What level of design work is needed?	10	
Individuality Did you copy code, modify it, and/or write it from scratch?	5	
Extra Credit (EC) ideas. Full credit ***only*** if the other parts of the code run well. Check with the teacher if *other ideas* qualify.		
Animating Actors with different images that work together as frames of animation.	+2	
Second player with a second set of keys/mouse. A special key for the other player. Multiple lives.	+2	
Exceptional craftsmanship in artwork and/or layout.	+2	
Total	100	

Option 2. Any Java Project.

Java Project ideas

To find ideas, google "Computer Programming Projects" "AP Computer Programming Projects" "Computer Science Programming Projects" "Java Programming Projects" "Computer Science Programming Projects"

Projects must incorporate topics from the previous chapters.
All students must use classes and objects (Chapters 8 and 9).
Students further along must also use a List, Set, Map, or recursion (Chapters 10–12).

- Games:
  - Battleship
  - Rush Hour
  - Tetris
  - Pacman
  - Sudoku (recursion. See textbook example, Chapter 12)
  - Checkers
  - Chess (no AI)

- Chapters 8–12 projects:
  - Chapter 8: Rational Numbers
  - Chapter 9: Critters
  - Chapter 10: Family Database
  - Chapter 11: Edit distance (or other project)
  - Chapter 12: Boggle (recursion)
  - Chapter 12: Towers of Hanoi (recursion)
  - Chapter 12: Anagrams (recursion)

- http://projecteuler.net/
  - Any project or projects of reasonable difficulty

- Simple calculator GUI
- Stop watch
- Game of life

Name: _____

		End-Year Final Java Project Rubric	Max	Points
		Required elements	[32]	
		If, else if, else		
		Loop (for, while, do while, or 'for-each')		
		Parameters		
		Methods, Return types (not just void)		
		Fields (instance variables)		
		Constructors that do something		
		Objects		
		new		
		Students beyond Chapter 8 must use one of: List, Set, Map, or recursion		
		More elements: Choose at least 5 of the following. Ask for possible extra credit for doing more.	[20]	
		Subclasses		
		array, List or ArrayList		
		Input from user		
		Input from file		
		Output to file		
		Inheritance (subclasses)		
		this		
		Explaining how borrowed code works		
		Graphics		
		Applet		
		GUI		
		Using Math library methods with reason		
		Sound		
		Collections		
		Casting		
		Difficulty and Complexity	[15]	
		Does it have complexity including many objects and methods? What level of design work is needed?		
		Overall ratings: Best 5 out of 8 (i.e. 3 will be dropped)	[30]	
		Meeting the Goal The program works.		
		Code Quality The class and method structure is organized and logical. Variable names make sense.		
		Comments and Java Doc Comments The programmer used appropriate comments and Java Doc comments in the code.		
		Development Process Key qualities: Planning (including explaining to teacher), effort, independence, tireless bug finding, challenging yourself at your level.		
		Individuality Did you copy code, modify it, and/or write it from scratch?		
		Timeliness Steady progress toward the goal		
		Positive Attitude		
		Helpfulness to other students		
		Choosing your project in a timely manner.	3	
		Extra Credit (EC). Full credit ***only*** if the other parts of the code run well. Check with the teacher if other ideas qualify.		
		Total	[100]	