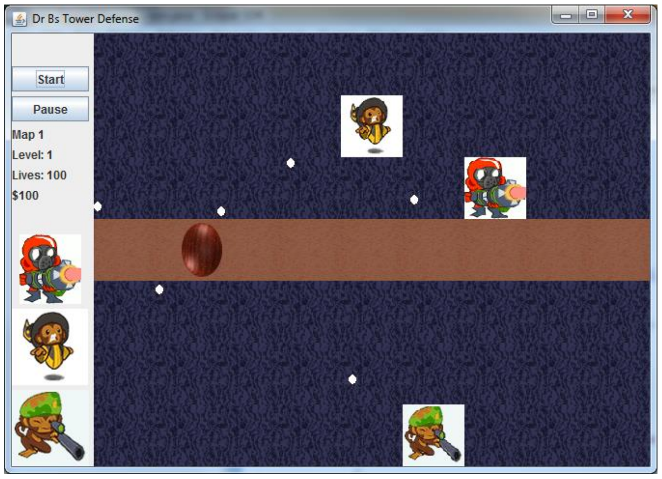
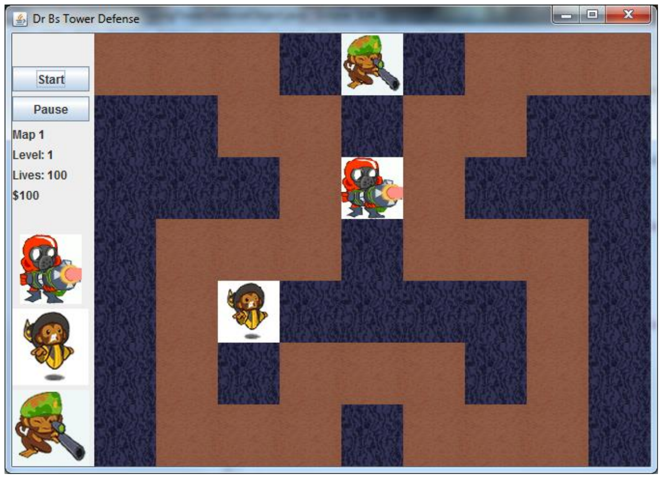
Tower Defense

We will be creating a tower defense game in various assignments during the rest of the semester. If you have never played a Tower Defense game, generally the user is allowed to place towers, which then stop enemies from making it to the end of a maze. The following is an example of what you might make.





TowerDefenseObject

Create a class called TowerDefenseObject. This class will be the parent class of most everything you see on the screen. Think about what instance variables you want, but you probably want at least the following information as instance variables

- An x and y position of an object.
  - You can use integers or doubles for this variable. Note that the map uses integers to place objects in particular locations, but things like bullets or arrows appear to fly more smoothly if you keep track of their location as a double.
- A BufferedImage.
  - This will be the image used to represent the object on the map.
- The image width and the image height as integers.
  - These ints will be used to draw the image so that it is scaled appropriately.

You should have at least one constructor for your TowerDefenseObject

- This constructor should probably take all the instance variables as parameters
- Note on the image width and height:
  - Ever BufferedImage knows the width and height as it exists on your computer. If you give a different width and height, then you can have the image be either larger or smaller than the standard and do not have to use an image editor to change the picture size.

You should have at least the following methods

- Getters and setters for all your instance variables (10? methods)
- Note that you will be using this object to create your game, so feel free to add any other methods you might want.

Polymorphism:

- You will likely want to code this up using polymorphism. As the picture of the object exists in this class, you probably want a method that can draw the image. We may not have gotten to this yet, so I suggest the following:
  - A method called 'drawTheImage' that takes a Graphics g as an argument. It should then use the following line of code to draw the image. Note that your instance variables may have different names than what I show below
    - g.drawImage(theImage,(int)theX, (int)theY,theImageWidth,theImageHeight,null);
      Note that you may only want to call this method if the image is not null

Final note: I might say some of these are optional, but check how the assignment will be graded so that you don't miss points. That checklist is at the end of this document.

Tower

Your Tower Defense game will have some type of Tower. Some of the information you want will be contained in the previous class you made, but be sure to create a class called Tower that extends TowerDefenseObject for these objects. Here are some suggested instance variables

- A shooting radius (you may or may not need this)
- Shooting: Think about what variables you might need to have in order for your towers to shoot. Are the towers going to constantly send out a barrage of 'bullets' or will there be a delay? To achieve a delay, you need at least the following 2 variables:
  - A variable that keeps track of how quickly the tower can reload and shoot again. For example, one tower might shoot 1 drop of water every 10 seconds and a different tower might shoot 10 fire arrows every 10 seconds.
  - A variable that keeps track of the time until the tower can shoot again. For example, there are 9 seconds left until the tower can shoot the 1 drop of water. The variable should probably starts off at 0, meaning an immediate shoot, then goes to the same number as the shooting or reload speed.

You should have at least 1 constructor.

- Probably an 8? Parameter constructor?
  - Be sure to setup all the instance variables correctly.
  - Don't forget to appropriately call the super constructor

You should have at least the following methods

- Getters and setters for the instance variables (6? methods)

Polymorphism and game time

- Each tick of the game represents 1 'picture' or 'snapshot. That means there is no for loop that draws everything repeatedly, but that every time your parents 'drawTheImage' method is called, your Tower sees 1 'unit' of time. If you want to count down time until your next show you need to override the drawTheImage method and update your own instance variables accordingly. Stated another way, every time this method is called you should likely decrements the time until the next shot by 1
- Don't forget, the drawing code is in your parent's 'drawTheImage' method. You don't have to draw the picture yourself if the first thing you do is call the parent's drawTheImage method.

Testing:

I have provided a class called MapWithTowers that you can use to test your constructors and see if the tower shows up on the screen in the correct location. You will need to fill in some of the information yourself to get it working.

- I have included a picture of myself as the 'Tower' and the BufferedImage should be setup and ready to pass to your constructors

MovingTowerDefenseObject

Some of your objects will move. Specifically, the Enemy and the Projectiles. Create a class called MovingTowerDefenseObject that extends TowerDefenseObject and that stores the following information as instance variables

- The velocity in the x direction and the velocity in the y direction. I would recommend these be stored as doubles to make the motion easier.

You should have at least 1 constructor

- I hope you are getting the hang of this now, but probably an 8 + 2 = 10 argument constructor. The 8 arguments to go the parent, and the 2 go to fill in your own instance variables.

You should have at least the following methods

- Getters and setters for your instance variables (4? methods)

Polymorphism

- Each time your moving object is drawn, it need to be drawn somewhere else the next time, or it won't move. To achieve this you can override the parent's drawTheImage method as follows
  - Call the parent's drawTheImage method so that the image is drawn
  - Change the X location (stored in the parent) using the X velocity. For example, if the object was at an x location of 5 and had a velocity of 2.0 in the X direction, the new X would be 7
  - Change the Y location using the Y velocity, similar to how you changed the X location.

Images

Create and add the following images to your project:

Note that the game we are creating will be played on a square based layout. 64 pixels by 64 pixels is an ok dimension.

- A road tile. (probably 64 by 64)
- A wall or non-road tile
- At least 2 enemies
- At least 2 towers
- At least 1 projectile (bullet, arrow, rock, dart, fire)

Enemy

Create a class called Enemy that extends MovingTowerDefenseObject and stores the following information as instance variables

- The health of the enemy

Note that we will assume enemies move along a straight line using an X and Y velocity.

You should have at least 1 constructor.

  - 1 constructor should be patterned after all the other constructors created thus far and set the instance variables appropriately.

You should have at least the following methods

- Getters and setters for the health of the enemy (2 methods)
- Enemies will be 'hit' by your projectiles. When they get hit, there needs to be a way to reduce their health. I would suggest a method for this.

Polymorphism:

- I don't know that Enemies need to do anything differently for any of the parent methods.

Testing:

I have provided a class called MapWithEnemy that you can use to test your constructors and see if the Enemies show up on the screen in the correct location and move with the x and y velocities you provide. You will need to fill in some of the information yourself to get it working.

- Don't forget to use your new images for the enemies

Projectile

Create a class called Projectile that extends MovingTowerDefenseObject and stores the following information as instance variables

- How much damage will the projectile inflict?
    - Will getting hit result in immediate 'death' or will some projectiles do less damage. You will need an instance variable to keep track of how much damage this projectile inflicts to make it a bit more challenging.
- We never really get to all the physics, so you might not want this variable, but projectiles should technically go the same velocity regardless of the direction.
    - This means a variable that keeps track of the total velocity the projectile has, and then you would use that number to 'update' the X and Y velocities in the parent. Again, you can get something nice working without this, so feel free to skip it.

You should have at least one constructor. I hope you can figure this out on your own now

- Add an optional copy constructor: Your towers are going to shoot many projectiles. A copy is easier to make when a constructor only needs one argument.

You should have at least the following methods

- Getters and setters for all instance variables (4 methods)
- A getDamage method.

Firing at enemies

- Somehow you are going to need aim these projectiles in the direction of your enemies. An easy way to do this is to let the projectiles figure out their own velocity based on which enemy you want them to hit. For example, the first enemy on screen.
    - A void fireAtEnemy(Enemy e) method.
        - This method does not have to work perfectly, but it should use the enemy's x and y location, the projectiles own x and y location, and then set the velocity X and velocity Y based on the velocity of the projectile.

Testing:

I have provided a class called MapWithProjectiles that you can use to test your projectiles and see if they will fire at an enemy. You will have to put the enemy and the projectile in different locations for different runs of your program to make sure the projectile goes every direction. I suggest that your enemy doesn't move for this test.

Update the Tower Class

- Towers should now 'have' a projectile, so add a projectile as an instance variable and change your constructor
- Add getters and setters for the projectile instance variables (2 methods)

Firing at enemies

- Just like the projectiles, the towers will be 'asked' to fire at an enemy. The think that will ask the tower to do that it probably your main game engine. That engine will need to be able to put the tower's projectile into an array so that the projectile can be drawn.
- Stated another way: The game engine will ask all the towers to fire at the enemies. It will expect those towers to give it projectiles if they can fire so that the game engine can draw all those projectiles. The towers won't want to give the game engine their only projectile, so they will need to give it a copy.
- Oh, and they might not be ready to fire yet, so they need to sometime not give it anything.
- Here is my idea:
    - Add a method that returns a Projectile called fireAtEnemy(Enemy e)
        - If the tower can't fire yet because the time until the next shot is greater than 0, this method should return null.
        - Otherwise,
            - the time until the next shot should be set to the shooting delay
            - You should create a new Projectile(give it your instance variable)
            - Then call the new Projectile's fireAtEnemy method, passing it Enemy e

Testing:

I have provided a class called MapWithTowersThatFire that you can use to test your towers and see if they will fire at an enemy. The MapWithTowersThatFire class only has 1 projectile in it currently, but hopefully your tower fires successfully

Upgrades you can do now:

- The MapWithTowersThatFire class only remembers the p1 Projectile. Upgrade the code to use an array of Projectiles so that the tower can fire multiple projectiles at the enemy.
- The MapWithTowersThatFire class only remembers the t1 Tower. Upgrade the code to use an array of Towers. Add a few more towers manually and see if they fire at the enemy
- The MapWithTowersThatFire class only remembers the e1 enemy. Upgrade the code to use an array of Enemies. Add a few more enemies manually.
- It is too early to figure out how to kill enemies and fire at different enemies, but that is coming soon.

Collision Detection

Remember all the projectiles

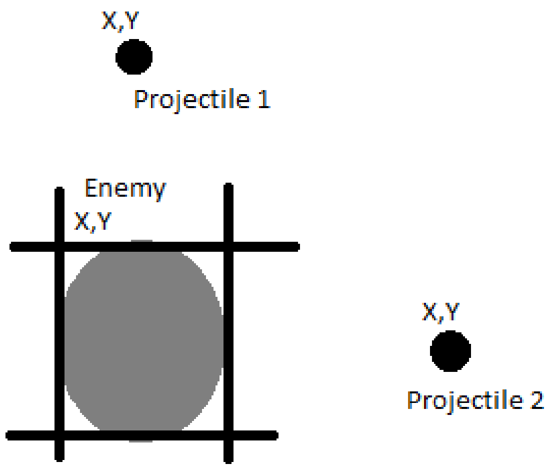
If you didn't complete the upgrades in the previous section, do them now. Specifically the ones with the projectiles. Use the MapWithTowersThatFire class as your starting point.

Although we added a tower's shooting radius instance variable, ignore that for now and just have each tower shoot at the first Enemy in your array.

- Make sure you have at least 1 tower firing multiple projectiles. One enemy works, but multiple enemies are required, so try to get multiple moving enemies.

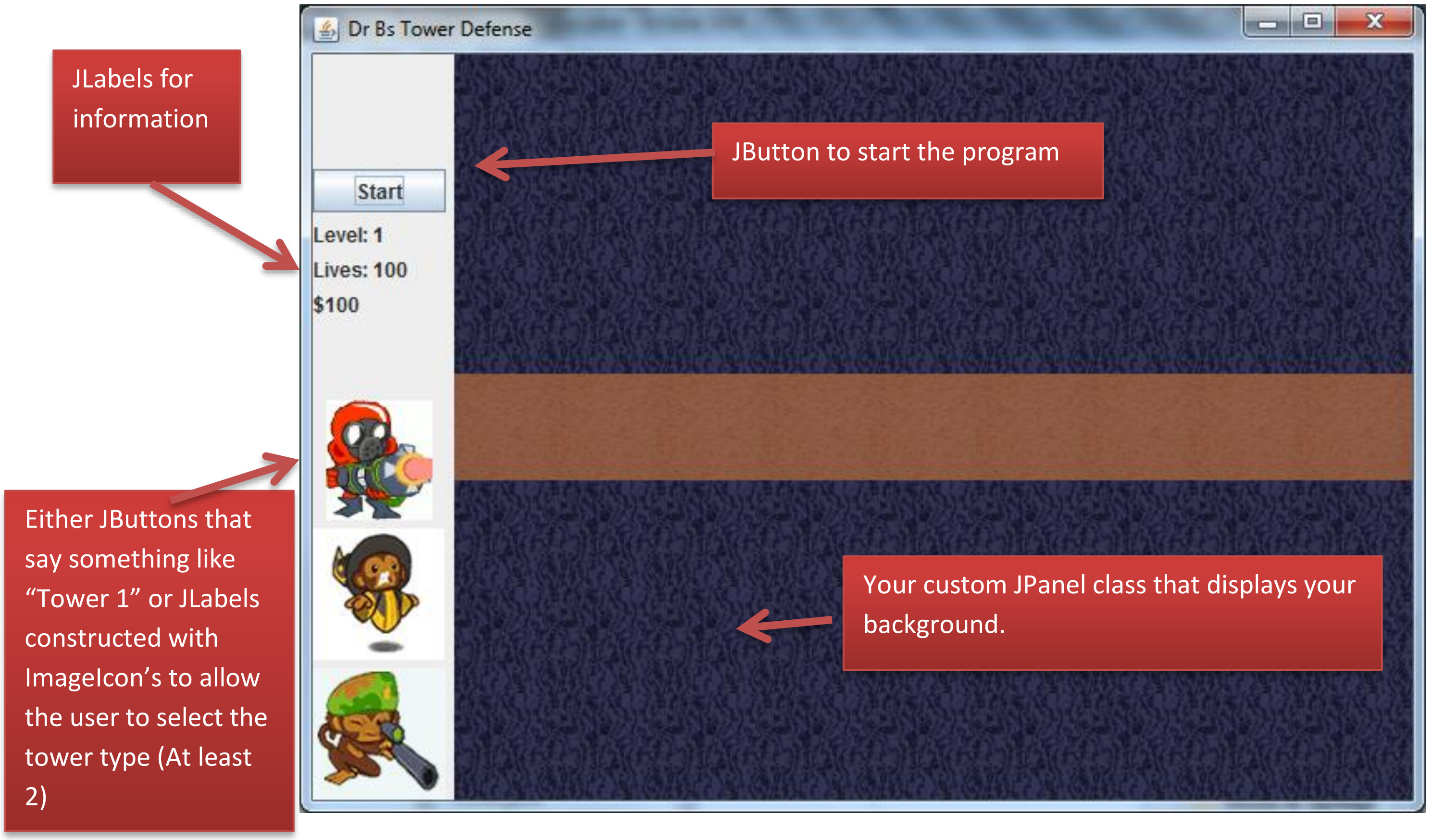
Get Collision Detection Working

Collision detection involves checking if the bullet is within the bounds of the enemy. For this you need some kind of for loop that looks at every enemy. Now say that you have an enemy called E. Now you need to check every Projectile to see if the projectile hit's this Enemy. Say that you have a Projectile called P. You now check if P's X and Y are within the bounds of the Enemy. If they are, then you reduce the enemies health by the damage of the Projectile. The following figure might help.



Your Gui / JFrame class

Make a class that extends JFrame that has the appropriate areas. It does not need to look like the figure below, but needs to have everything shown. Also, don't be tempted to start with the MapWithEnemy.java or MapWithTower.java file as those files are not a good starting place for a well-built GUI.


JLabels for information
Dr Bs Tower Defense
JButton to start the program
Start
Level: 1
Lives: 100
$100
Either JButtons that say something like "Tower 1" or JLabels constructed with ImageIcon's to allow the user to select the tower type (At least 2)
Your custom JPanel class that displays your background.

Your JPanel Class

Make a separate class that extends JPanel. Add this class as an instance variable to your JFrame class. This class should

- Load a map with a single straight path to the right in the middle using your images.
  - You can just use a single picture for the background. Stick that picture into a BufferedImage variable and draw it in the paintComponent code as explained below.
  - You can use many little pictures and setup a grid of pictures. You can read in the information from a text file as we did in the FileReader assignment, or just hard code it in to look like this. Probably do this in your constructor and have a BufferedImage[][] to save the images into, and even perhaps a String[][] just to keep the map information.
- In the following method, draw each of your images

  -
    ```
    @Override
    public void paintComponent(Graphics g){
      //code here to draw your image or images
    }
    ```

Brains?

We now start on the ‘brains’ of Tower Defense. These should be put into your JPanel class. You should have already created some of the brains in the MapWithTowersThatFire class. You can copy your code from that class into your JPanel class.

Get Enemies Working

- Add an array of Enemy’s, or TowerDefenseObjects if you want to be more general to your JPanel probably as private instance variables.
- Once you have that, add a startGame method to your JPanel and add an ActionListener to your Start button that calls the startGame method when you click it.
- Then, when the user clicks the start button, your startGame method should add some enemies to your array, and set their location to be to the left of the screen with a positive X velocity and a Y velocity of 0
- Adjust your JPanel constructor so that it now requires a JLabel for the lives and also has an instance variable that keeps track of how many lives you currently have. You will then need to fix your JFrame so that it passes the correct JLabel to the JPanel constructor.
- Once an enemy is off the screen to the right (check the X location), remove it from the array, decrease the number of lives by the health of the enemy, and call the setText method of the JLabel so that it now shows fewer lives.

Mouse Motion Listener

Get placing a tower and drawing all the towers placed working

- Add a MouseMotionListener to your JPanel. In the mouseMoved method, change some instance variables so that the JPanel can know about the mouse x and y locations, and then call repaint() so that the JPanel can repaint when the mouse is moved.
    - Once that is working, add a placeTower(int towerType) to your JPanel. This method should change an instance variable. Then in your paintComponent method, add an if statement to draw the tower the user selected at the location of the mouse based on the instance variable.
- Add an actionListener to the JButton or JLabel in the JFrame, and have it call the placeTower method of the JPanel when clicked. The tower should now follow the mouse when the user click’s it on your GUI.
- Get the JPanel’s mouseClicked listener to add a tower to an array of Tower’s or to your TowerDefenseObjects array if you decided to go that route. Make sure to call repaint() and also to call the drawTheImage method of any item in your array.
- Decrease the amount of money the user has once they place the tower

Other things I forgot

- Add anything else to get the game working that I forgot to mention.
- Enjoy playing your game

Tower Defense Checklist

Total: _____/9

Towers **Name: ___________________**

________ There are at least 2 towers the user can add.
________ Towers can be added to the map in locations the user selects.
________ When adding a tower, the tower image follows the mouse until the user selects the proper spot.
________ After adding a tower, the user's money is reduced, and the money label is updated.

Enemies

________ When the Start button is clicked, at least 5 enemies walk across the screen and the enemies are not all piled on top of each other.
________ When an enemy reaches the end and has not been defeated, the user's total lives are reduced and the lives label is updated.

Projectiles

________ The towers fire at the enemies
________ If a projectile hits an enemy, the enemy's health is reduced.
________ An enemy can lose all their health and not make it to the end.