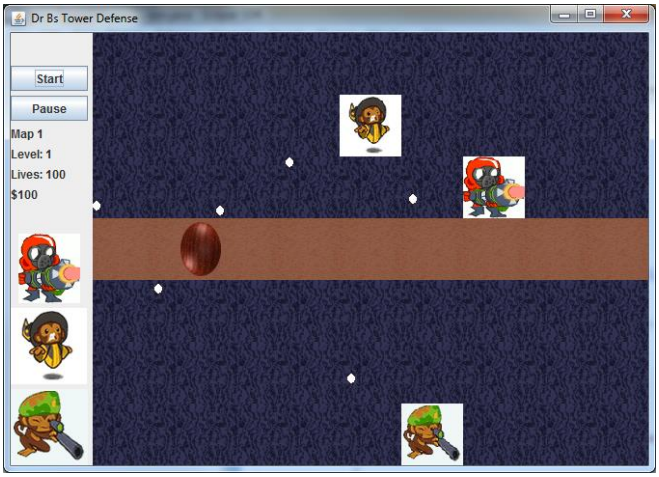
Tower Defense

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 that stores 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 the following 2 constructors

- A constructor that takes an x, y, the BufferedImage, the image width and image height.
- A constructor that takes the x, y and the BufferedImage. You can get the width and height for your instance variables directly from the BufferedImage so that no scaling occurs.

You should have at least the following methods

- Getters and setters for all your instance variables (10 methods)
- A method called drawTheImage that takes a Graphics g as an argument. It should then use the following method to draw the image. Note that your instance variables may have different names than what is 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
- Note that you will be using this object to create your game, so feel free to add any other methods you might want.

Tower

Create a class called Tower that extends TowerDefenseObject and stores the following information as instance variables

- A shooting radius (you may or may not need this)
- 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 2 constructors to mimic the parent class. Hopefully you now know what should be in a constructor, meaning what information you need to give to the parent, and what extra information you need to fill your own instance variables.

- Both constructors should set the instance variables appropriately.
    - This should call the appropriate super constructor

You should have at least the following methods

- Getters and setters for the instance variables (6 methods)
- Override the drawTheImage method so that it calls the parent's drawTheImage method and also decrements the time until the next shot by 1

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.

- Make sure to test both constructors.
    - The one where the image width and height are not provided, and hopefully the image is properly proportioned.
    - The one where you provide a width and height. The image can be squished in either direction depending on the numbers you provide.

MovingTowerDefenseObject

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 stored as doubles

You should have at least the following 2 constructors

- The same 2 constructors as the parent, with the addition of the 2 doubles for velocity at the end of the list. Make sure that you call the parent class' constructors appropriately.

You should have at least the following methods

- Getters and setters for your instance variables (4 methods)
- 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 2 constructors.

- The first constructor should probably accept the x, y, health, velocity x, velocity y, and image
    - This should call the appropriate super constructor
- The second should probably accept everything as above as well as an image width and image height.
    - This should call the appropriate super constructor

You should have at least the following methods

- Getters and setters for the health of the enemy (2 methods)
- A method that decreases the health by an amount for when the enemy is hit with a projectile.

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.

- Use your new images for the enemies
- Make sure to test both constructors.
    - The one where the image width and height are not provided, and hopefully the image is properly proportioned.
    - The one where you provide a width and height. The image can be squished in either direction depending on the numbers you provide.

Projectile

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

- The damage the projectile inflicts
- The velocity of the projectile. Note that you will figure out the x and y velocity based off of this velocity.

You should have at least 3 constructors, with the addition of a new one mentioned below. Hopefully you now know what should be in a constructor.

- 2 constructors should be patterned after all the other constructors created thus far and set the instance variables appropriately.
    - This should call the appropriate super constructor
- 1 constructor should take a Projectile p and then set the instance variables based off of P's instance variables. For example,
  this.damage = P.damage;
    - public Projectile(Projectile p){
      // code here
      }
  This new constructor will be used in the Tower class. It is as if the Tower class had a projectile and then made copies of it and fired these copies at the enemies. A copy is easier to make when a constructor only needs 1 argument.

You should have at least the following methods

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

Update the Tower Class

- Add a projectile as an instance variable
- Add getters and setters for the projectile instance variables (2 methods)
- 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: You will need the GUI first, which you will make in the next assignment

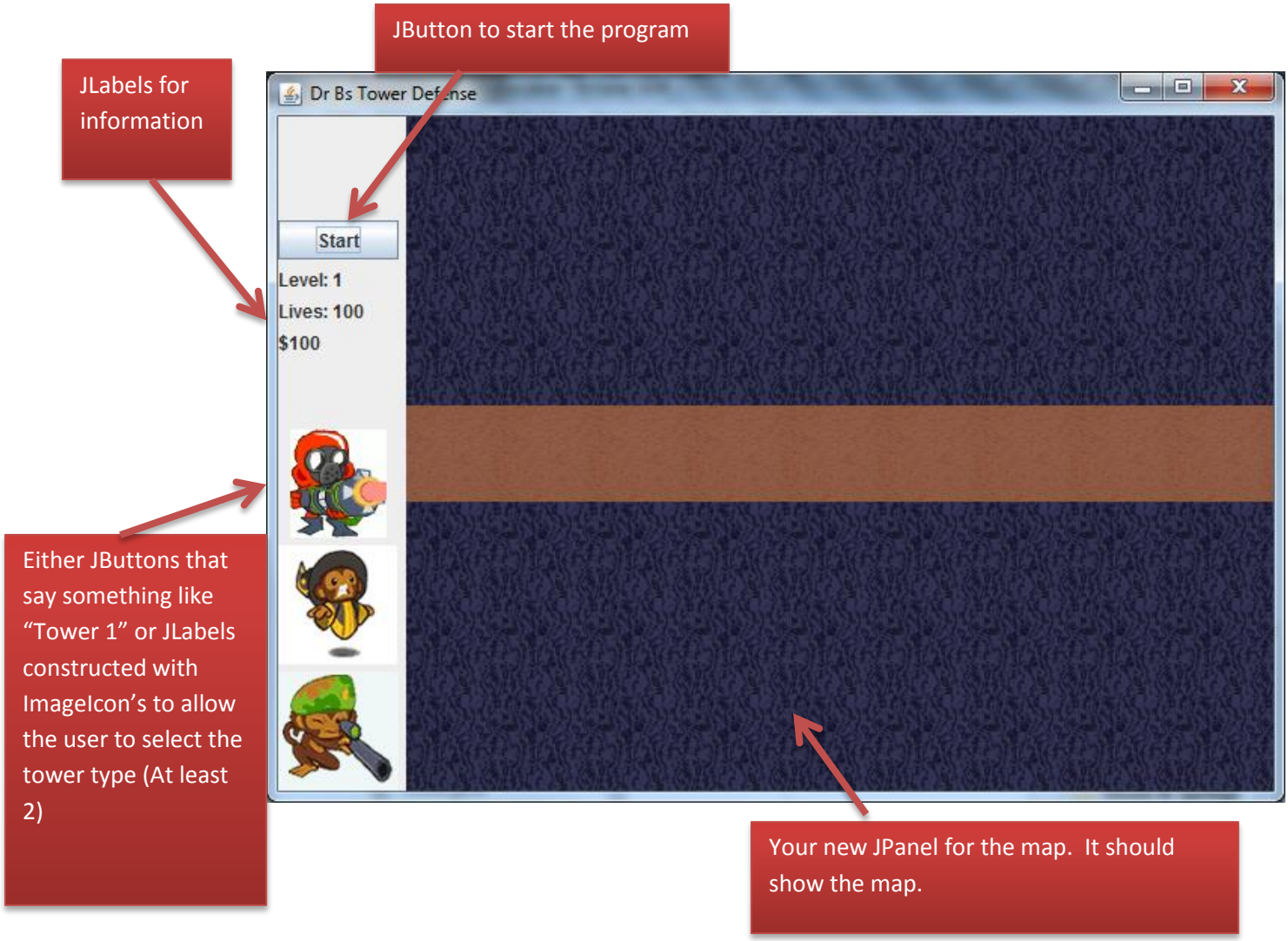
YourSuperAwesomeJFrame

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. 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.

JPanel

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 read in the information from a text file, 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 each of the images
      }



Brains?

We now start on the 'brains' of Tower Defense. These should be put 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
- To get the enemies moving you need to update the paintComponent method to:
    - call the drawTheImage method of every enemy that is in your array.
    - add the following code:

```
repaint(); //so that the paintComponent method will be called again – like a loop, but not quite
try {
        Thread.sleep(10); //so that things do not happen too fast.  You can change this number
}catch (InterruptedException e) {
                e.printStackTrace();
}
```

    Your enemies should now walk to the right of the screen.
- 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

Projectiles Firing and Collision Detection

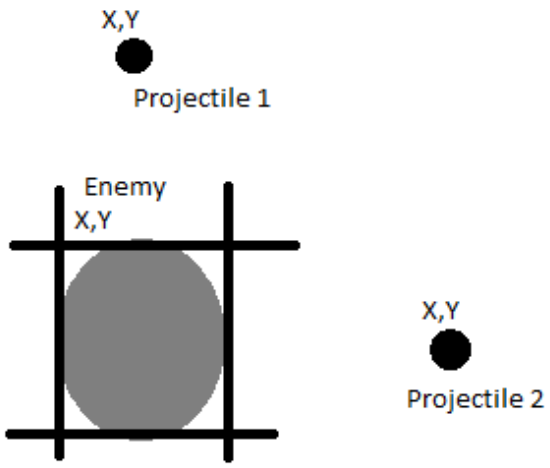
Get Projectiles Firing

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.

- Add a Projectile array to your JPanel, or use the TowerDefenseObjects array if you decided to go that route.
- Call the drawTheImage method of all your projectiles.
- When you are drawing your towers, if the tower canFire(), then call the fireAtEnemy(the first enemy) method. This method returns a Projectile. Add that projectile to your array and projectiles should now be working.

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.



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.