CECS 277 – Project 5 – Oregon Trail Hunter

Create an animated GUI application that recreates the hunting game from Oregon Trail.

You can play a version of the game here. Start on the trail, then choose option 8 on the situation menu to hunt: https://archive.org/details/msdos_Oregon_Trail_The_1990

1. Create an abstract Entity class
    a. Rectangle location – this is a class in java.awt, it has an x, y location and a width and height. It also has handy functions like translate() for moving the object and contains() for testing if another point is contained within it.
    b. int hp – the number of hit points an entity has.
    c. int speed – speed the entity travels at
    d. boolean moving – whether the entity is moving at this time
    e. int direction – which direction the entity is moving in. This is defined by the eight cardinal directions (N, NE, E, SE, S, SW, W, NW (values 1-8)).
    f. Entity has many methods listed in the class diagram, the update() method should call move() and draw(). Each class that extends Entity will be forced to override the draw() method.
2. Create a Hunter Class
    a. Bullet bullet – the hunter's projectile. The hunter can only fire one shot at a time. If a second shot is fired while the first one is still in flight, the previous one is replaced with the new one. If you want your game to be able to have multiple shots in flight at the same time, change this into an ArrayList of Bullets. Remember to remove bullets from the list after they've left the edge of the screen, otherwise they continue to exist (ie. take up memory).
    b. A hunter has a method to fire a bullet, and a method to test to see if the bullet hit a target.
3. Create a Bullet Class
    a. Add a method to test to see if it has hit a target.
4. Create a Quarry (creature to hunt), and a QuarryGenerator class
    a. Have the QuarryGenerator read in a file of quarries to hunt.
        i. Name, weight, hp, and speed
        ii. Based on the weight, increase the size of the quarry's bounding box, and decrease their speed.
5. Create an Obstacle Class
    a. An obstacle has a type
        i. Grass, bushes, trees, fallen quarries, etc.
    b. Obstacles have a method to test to see if another entity has collided with it.
6. Create a Panel class – extends from JPanel implements KeyListener, MouseListener, and MouseMotionListener
    a. Set the background color to black.
    b. Make a Hunter, QuarryGenerator, and ArrayLists of Quarries, and Obstacles.
    c. Randomly assign the number and locations of obstacles. Store them in the arraylist to make it easy to iterate though to test for collisions.
    d. A Thread to refresh the screen ~every 50ms, call repaint(), test for collisions between the entities, and randomly release a new Quarry.
    e. MouseMoved() and MouseClicked() - User should be able to point their cursor in the direction they want the hunter to face (one of the eight cardinal

directions). (Use the equations for a right triangle to calculate this). When they click the mouse button the hunter fires a shot.

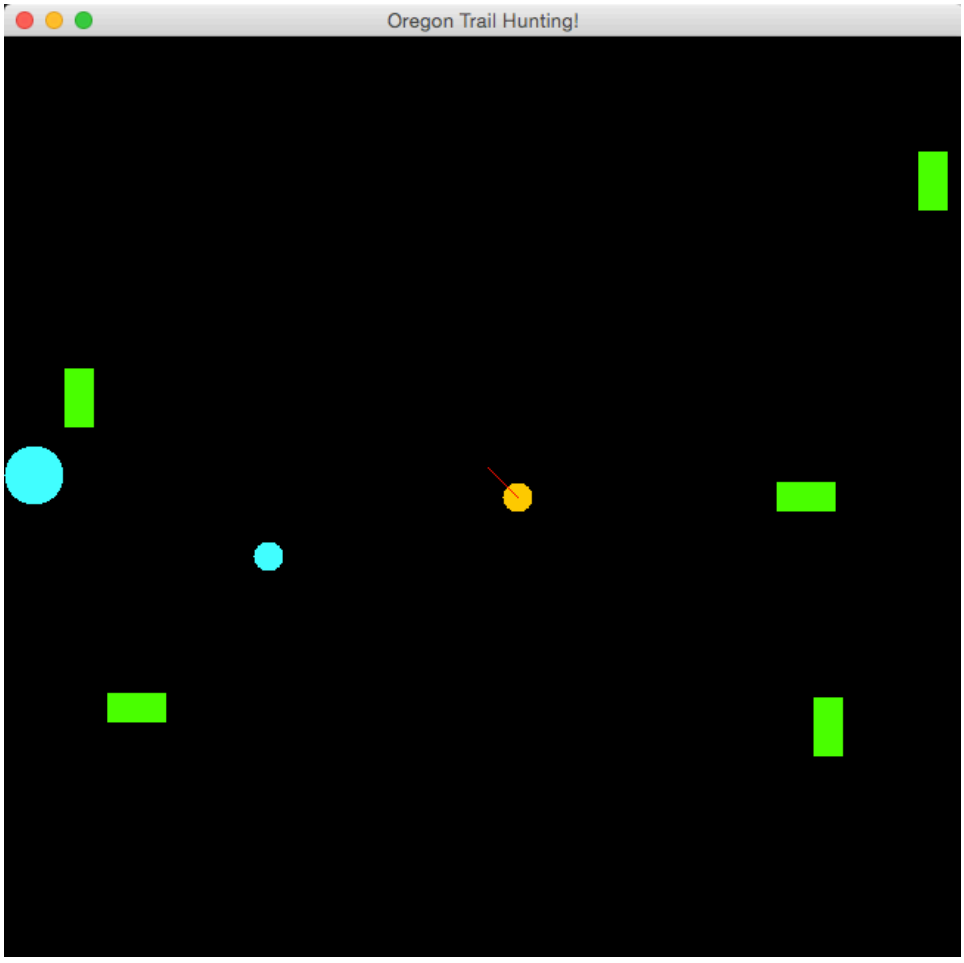
f. KeyPressed() – User may press keys the left and right arrow keys to spin clockwise and counter clockwise respectively. Press the Enter key to start and stop moving in the direction they are facing. Press spacebar to fire a shot.
g. paintComponent(Graphics g) – This is where you call your update method for each of your entities.

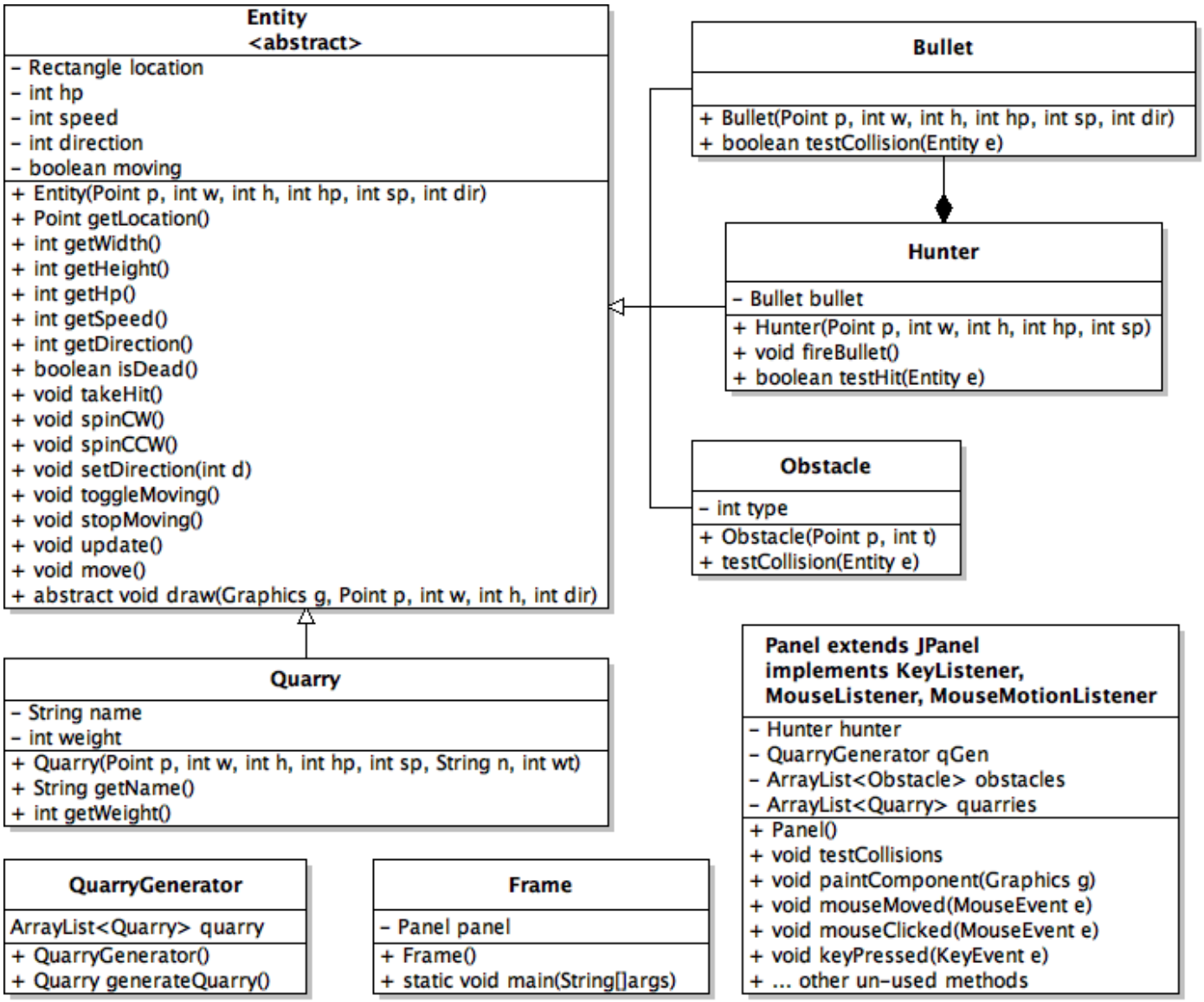
7. Create a Frame – extend from JFrame
    a. Set an appropriate window size for the computer's resolution.
    b. Declare your panel.
8. Basic Gameplay – your game must follow these rules.
    a. The hunter may move at a constant rate in any of the cardinal directions by using the arrow keys and hitting the enter key. If the hunter is moving when an arrow key is pressed, the hunter will change directions without stopping.
    b. The above is also true for mouse movements.
    c. The hunter may not move through any obstacles. If the hunter runs into an obstacle, the hunter immediately comes to a stop.
        i. Trees, boulders, bushes, fallen quarries, and the walls of the screen are all considered as obstacles for the hunter.
    d. The user may either click the mouse or press the space bar to fire a shot.
        i. When a hunter fires a shot, it travels across the screen in the same direction that the hunter was facing. If the hunter turns while the bullet is traveling, the bullet should not turn also.
    e. Bullets are affected by the same obstacles as the hunter.
    f. Quarries are released at the edge of the screen and run in a random direction. If they encounter an obstacle they randomly choose another direction and run that way. This repeats until they exit the edge of the screen.
        i. When they exit the screen they are removed from the quarries list.
        ii. The edge of the screen is not considered an obstacle to quarries.
    g. If the hunter hits a quarry, the quarry is killed and becomes an obstacle.
9. Variable Gameplay – choose the goal of your game.
    a. Time based – what you shoot in the given time is what you receive.
    b. Goal based – must bring home a set amount of meat.
    c. Round ends after a set number of bullets have been fired.
    d. Percent hits – keep track of the number of misses/hits. Pass to the next level if the correct percentage is reached.
    e. Have a shop where you can sell the meat and buy more bullets to move on to the next level.
    f. Hunter is mauled by bears, trampled by bison, or bitten by squirrels and dies.
    g. Combinations of these, or something else entirely.
    h. If you're anti-hunting, pretend you're throwing hearts at stuffed animals?
    i. You may need to add additional variables/methods to accomplish these tasks.
    j. You may need to add dialog boxes to get input from the user. The console input will still work, but should be avoided.
    k. Quarry hp is suggested, not required. If you want different levels of difficulty, some quarries could need to be hit multiple times.
    l. You may modify the quarryList.txt file to add new items, change values, etc.

10. Hints:
    a. Start small and build up from there. Test it after adding something new:
        i. Get the frame and panel working. Change the size, background color.
        ii. Add a Hunter. Start with just drawing a circle for the hunter, and a line representing the gun.
        iii. Add the Thread, have it call repaint().
        iv. Add the keyPressed(), add the arrow keys and get the hunter to rotate.
        v. Add the enter key to the keyPressed() to get the hunter to move.
        vi. Add the Bullet and the space bar key to fire the gun.
        vii. Add the mouseMoved and click to move and fire the gun.
        viii. Add the obstacles, draw green rectangles to the screen.
        ix. Add the collision detection to check if the hunter ran into an obstacle.
        x. Add the quarries. Draw circles and have them run around the screen.
        xi. Add the collision detection for the quarries. Obstacles and bullets.
        xii. Add the finishing touches. Gameplay ideas, drawing detail, etc.
    b. Use the functions in Rectangle to do most of your collision detection for you. You can pass in a point or rectangle to see if two entities have collided.
    c. If your hunter's rectangle has different height and width, it may get stuck on obstacles when changing direction, it's easier if you make it a square by keeping the width and height the same. Change later if you want to add detail.
    d. The console still works. If you're stuck, print out values to debug problems.
11. Extra Credit:
    a. Use images/sprites to represent your quarries, hunter, and obstacles. Add the name of the image file to the quarryList.txt file.
    b. Animate your hunter and quarries. Have different poses for the different directions or actions of your sprites.
    c. Add sounds to your game.

Original game screen caps:

Rough Implementation (before drawing detail is added):




Entity
<abstract>
- Rectangle location
- int hp
- int speed
- int direction
- boolean moving
+ Entity(Point p, int w, int h, int hp, int sp, int dir)
+ Point getLocation()
+ int getWidth()
+ int getHeight()
+ int getHp()
+ int getSpeed()
+ int getDirection()
+ boolean isDead()
+ void takeHit()
+ void spinCW()
+ void spinCCW()
+ void setDirection(int d)
+ void toggleMoving()
+ void stopMoving()
+ void update()
+ void move()
+ abstract void draw(Graphics g, Point p, int w, int h, int dir)
Bullet
+ Bullet(Point p, int w, int h, int hp, int sp, int dir)
+ boolean testCollision(Entity e)
Hunter
- Bullet bullet
+ Hunter(Point p, int w, int h, int hp, int sp)
+ void fireBullet()
+ boolean testHit(Entity e)
Obstacle
- int type
+ Obstacle(Point p, int t)
+ testCollision(Entity e)
Quarry
- String name
- int weight
+ Quarry(Point p, int w, int h, int hp, int sp, String n, int wt)
+ String getName()
+ int getWeight()
Panel extends JPanel
implements KeyListener,
MouseListener, MouseMotionListener
- Hunter hunter
- QuarryGenerator qGen
- ArrayList<Obstacle> obstacles
- ArrayList<Quarry> quarries
+ Panel()
+ void testCollisions
+ void paintComponent(Graphics g)
+ void mouseMoved(MouseEvent e)
+ void mouseClicked(MouseEvent e)
+ void keyPressed(KeyEvent e)
+ ... other un-used methods
QuarryGenerator
ArrayList<Quarry> quarry
+ QuarryGenerator()
+ Quarry generateQuarry()
Frame
- Panel panel
+ Frame()
+ static void main(String[]args)