1. Nonplayer
2. Laser
3. Linear Enemies
4. Main
5. Player
6. Text files for the levels
7. Game
8. Panel
9. Frame
10. World
11. wall
12. goal

WHAT WE GOTTA WORK ON: How do the non players know their direction?
How does reading from the text file work?
The constructor for linearEnemy

1. public class NonPlayer
    a. private int direction;
    b. private int type;
    c. private int gridX;
    d. private int gridY;
    e. private int speed
    f. private Rect rectangular
    g. public updateRectangle() - updates the rectangle of the nonplayer
    h. public NonPlayer(int direction, int type, int gridX, int gridY, int width, int height) - sets the direction to a randomly generated int, type, gridX and gridY to the received values, sets the rectangle to a new rectangle with constants such as width and height specific to each nonplayer type
    i. public int getType() - will return a number corresponding to the type of Nonplayer that this is (ie laser, linear mover, box mover, etc)
    j. public int getDirection() - will return a number (1 - horizontal, 2 - vertical, 3 - Box) that indicates the direction the Nonplayer performs its attack/movement
    k. public int getGridX() - will return the x value of the Nonplayer in the level grid
    l. public int getGridY() - will return the y value of the Nonplayer in the level grid
2. public class Wall extends NonPlayer
    a. public Wall(int direction, int type, int gridX, int gridY, int width, int height) - calls the super constructor with the constants for width and height for a wall
3. public class Goal extends NonPlayer
    a. public Goal(int direction, int type, int gridX, int gridY, int width, int height) - calls the super constructor with the constants for width and height for a goal
4. public class Laser extends Nonplayer
    a. private Rectangle beam - the laser beam that kills the player
    b. private int charge - holds the charge of the laser between 0 and 100
    c. private Color laserColor - the color of the laser at any point based on changing RGB values based on laser charge
    d. public void chargeLaser() - adds charge to the laser
    e. public Laser(int direction, int type, int gridX, int gridY, int width, int height) - calls the super constructor with the constants for width and height for a Laser (the body of the laser emitter, not the beam)
5. public class linearEnemy extends Nonplayer
    a. public linearEnemy(Rectangle rect, int moveDist, int direction, int gridX, int gridY, int width, int height) - calls super and gets listed values, x and y taken from rectangle
    b. private int moveDist - how far up and down or left and right the object will move

    c. public void mover() - changes the Nonplayer’s location according to its move patterns
6. public class patternEnemy extends Nonplayer
    a. private int currentDirection - the direction it is currently moving
    b. public linearEnemy(Rectangle rect, int moveDist, int direction, int gridX, int gridY, int width, int height) - calls super and gets listed values, x and y taken from rectangle
    c. private int moveDist - how far up and down or left and right the object will move
    d. public void mover() - changes the Nonplayer’s location according to its move patterns as well as changing the current Direction when appropriate, achieving a square pattern
7. public class main
    a. public static void main(String[] args) - creates the frame
8. public class Panel extends JPanel implements KeyListener, Runnable
    a. public Panel()
    b. a key listener
    c. public void paint(Graphics g)
    d. public void reset()
    e. private World world
    f. private Player player
    g. private Thread t
    h. private currWorld
    i. private ArrayList<World> worlds
    j. private int updatesPerSecond;
    k. public void update() - updates the game, calls the beatLevel on the world object and increases the currWorld if true
9. public class Frame extends Jframe
    a. Frame(String FrameName)
    b. Panel p
10. public class Player
    a. private int gridX - initial grid location of the player’s x coordinate
    b. private int gridY - initial grid location of the player’s y coordinate
    c. private Rectangle hitbox - the hitbox of the player
    d. private int direction - the direction of the player
    e. private int lives
11. World
    a. public World(File f) - constructor
    b. private ArrayList<Nonplayer> NonplayerList - all enemies on the level
    c. public Nonplayer[][] textConverter() - converts the text files representing the worlds into actual worlds
    d. public void die() - kills the player
    e. public void checkInteraction() - checks to see if the player is touching anything and then reacts accordingly
    f. public void levelWon() - checks if level is won

   g. private File f - file we will read the text from
   h. private Scanner fileReader(f) - reads the text files for the levels

12. Constants
   a. public static final int LINEAR_SPEED - 10
   b. public static final int PATTERN_SPEED- 5
   c. public static final int PLAYER_SPEED - 10
   d. public static final int UP - 1
   e. public static final int DOWN- 2
   f. public static final int LEFT - 3
   g. public static final int RIGHT - 4
   h. public static final int LINEAR_ENEMY- 1
   i. public static final int PATTERN_ENEMY- 2
   j. public static final int LASER - 3
   k. public static final int WALL - 4
   l. public static final int GOAL - 5
   m. public static final int PLAYER - 6