Updated Program Specification

Our project will be a game called Burst Your Bubble. The player controls a character called Mr. Burster. The job of the player is to shoot bubbles (spheres) that enter the screen with random velocities, sizes and locations. The player should avoid being hit by the bubbles. Each bubble has a number of points and health. And once the bubble reaches the health level zero, the player gets the points of the bubble added to his score. The score is represented in the top right corner of the screen and the level is represented in the middle of the screen.

How will the user input work?

The player will move around using “a” for left and “d” for right. Shooting is on at all times.

How will the program respond?

The player is a sphere that moves across the bottom of the screen in accordance with the direction the user inputs. Bullets will be shot up vertically from player.

What purpose does it serve?

The program is supposed to be a simple game for entertainment.

Namely, provide a list of the classes you will define.

Classes: Main, Runner, Player, Sphere, Bullet

Class: Runner

1. Describe the semantics and use of the class. What does it represent? When your program is run, does one instance exist? or a few? or many? Runner is inside of the Game class. Once a game object is constructed in the main method of the game class, the thread gets a runner object as a parameter and starts.

2. What are the member variables (names and types)? Runner won’t take in any variables.

    What does each represent semantically? N/A

    Why are they public/private? N/A

    Why are they the type they are? N/A

3. What the constructor(s)?

    If there is only one, why? There will be no constructor, since all it does it start the game

    If there is more than one, why? and how do they differ? N/A

4. What are the methods (return type, name, argument types)? One method: public void run() 1. What do they do (in words, not code)? it handles updating the screen, collision detection, updating the score, removing dead spheres and reprinting the screen.

    Why are they public/private/static/not-static? run() is public so it can be called in the Main class.

    Are they recursive? No

Class: Main

1.Describe the semantics and use of the class. What does it represent? When your program is run, does one instance exist? or a few? or many?

Main contains everything, similar to the world class in other projects. code heavy portion of the project, and it will run once. It'll have only one instance of it created.

2. What are the member variables (names and types)?

public static final int width = 1024; //width of screen

public static final int height = 750; //height of scree

public static final int FPS = 120;// number of Frames per second the game shall run on

public Player player; //the representation of the player playing the game

public Timer timer = new Timer(); // a Timer that time certain events, such as a brief pause after clearing all spheres

public Vector<Sphere> spheres;//a Vector all spheres shall be contained in to be updated

public int score; //the overall score of the game

public int level;// the level the player is on

private String scoreString;// the visual representation of the score

public String levelString;//the visual representation of the level

public String message;// this will display a message tell the player if they won or lost

int count = 0;

boolean isPlaying = true; //determines whether the game is being played or if it's on the start or game over screen

boolean isShooting = true;// determines wheter the player is shooting or not

public BufferedImage desert; // desert background

public BufferedImage city;// city background

public BufferedImage ocean;// ocean background

public BufferedImage space;// space background

public BufferedImage title;// title screen (from Spongebob!)

What does each represent semantically? Player is the object that the user will control. The vector of spheres is going to be the spheres entering the screen. The score represents the player's current score. scoreString represents the score in a string format so it can be drawn on the screen. The height and width represent the height and width of the playscreen. FPS represents the frames per second. Rand is a random object used for generating random numbers in the program. Timer is used to create tasks that happen at certain intervals. Buffered images represent the background images in game.

Why are they public/private? The score and scoreString are the only private variables because it should not be tampered with outside the context of world. The other variables are directly accessed and updated and they are public.

Why are they the type they are? Player is a player, that will have its own defined variables and methods. We choose an vector of Spheres because this game is going to be an endless shooter, meaning a vector would be helpful for continually spawn spheres. Score is an int because it will be incremented by whole integers. Height and width are integers because JFrame takes in ints to set the dimensions of the frame. Rand is a random object because we need a random object for generating random numbers.

3. What the constructor(s)? Main will only have one constructor.

```
public Main(){

    addKeyListener(this); //allows key inputs to be recorded

    spheres = new Vector<Sphere>();

    player = new Player();

    scoreString = "" + score + "";

    levelString = "";

    message = "Press space to start";

    Thread mainThread = new Thread(new Runner());

    mainThread.start();

    score = 0;

    level = 0;}
```

If there is only one, why? There only needs to be one constructor since there's only one version of the game that can be played.

If there is more than one, why? and how do they differ? N/A

4. What are the methods (return type, name, argument types)? private void bounce( ), private void draw(), private void update( ), public int updateScore( ), private void displayScore( ), private void collide( ), public void displayStartScreen, public void GameOver( ), private void spawn( ), public void restart( )

What do they do (in words, not code)? Bounce makes sure the spheres bounce of the sides of the playscreen. Draw will draw the player and the spheres to the screen and update will update the positions of the player and the spheres while the game is in action. UpdateScore will update the numerical value of score while the game is running based of the player's actions and displayScore will display that score at the top of the screen. Restart will restart the game. DisplayStartScreen will happen when the game is booted up for the first time, showing the name of the game and directions on how to start the game. GameOver will be when the player loses all of their health. On the game over screen, there will be a prompt to restart the game.

Why are they public/private/static/not-static? The private methods are private because they most likely will not be used outside the context of game. Methods like bounce, spawn, update, etc are exclusive to world because they are things that happen in the world. Restart, GameOver, StartScreen affect everything, so it's probably the best idea to leave them public for now.

Are they recursive? None will be recursive.

Class: Player

1.Describe the semantics and use of the class. What does it represent? When your program is run, does one instance exist? or a few? or many?

The player class represents the controllable player on screen. The player should be able to move left and right using A and D respectively, and should be able to autofire bullets with the space key or W. When the program is run, one instance of Player exists at first, but once a new level is started, a new instance of Player is created to reset their position.

2. What are the member variables (names and types)?

What does each represent semantically? Why are they public/private?

Why are they the type they are?

Name and Type	Notes
private int health	Health represents the “amount of time” the player can be in contact with an enemy Sphere before dying. This variable is private because it is only altered within the Player class, so there’s no need for it to be accessible to the other classes. Health is an int because a number is an easy to understand way to represent health, since the more the number depletes, the more danger the player is in.
private String healthDisplay	HealthDisplay is the visual indicator of what the player’s health actually is. The variable is private because it’s only used in the Player class, so making it public only draws the possibility of it being changed. This variable is a String because that’s the requirement for it to be utilized in the Graphics method drawString.
private boolean hasBeenHit	HasBeenHit detects whether the player is in contact with a sphere or not. If player is in contact with a sphere, hasBeenHit will be true, and the decreaseHealth method would start (hitDetection and decreaseHealth are used in tandem in Main). HasBeenHit is only changed in the player class, so we made it private. HasBeenHit is a boolean because booleans are a great way to describe something with two states. In this case, the two states are “The player is safe” or “The player is being hit”.
private double centerX	CenterX represents the x coordinate of the center of Player. This variable is used to set the position of player and is used when calculating if Player’s in contact with a sphere. It’s private because it’s only used in the Player class. It’s a double so the collision calculations can be more precise.
private double centerY	CenterX represents the y coordinate of the center of Player. This variable is used to set the position of player and is used when calculating if Player’s in contact with a sphere. It’s private because it’s only used in the Player class. It’s a double so the collision calculations can be more precise.
public double velocityX	VelocityX just dictates whether to move the player left or right. The value of velocityX is added to centerX to do this. This variable is public because it’s used in the KeyEvent methods in Main. This variable is a double because it will be added to another double, centerX.
private boolean dead	Dead detects whether the Player’s health has reached 0. Boolean dead triggers the method isDead, and isDead determines whether the game is over or not. The boolean dead is private because it’s not used outside of the Player class. Booleans are good for two-state scenarios. Dead covers if they player has above or below 0 health.
private double radius	Radius is meant to keep track of the distance between the center of Player to any edge of the Player’s circular body. This value is private because it’s not used outside of Player. We chose this variable to be a double for precision in the collision equations.
public Vector<Bullet> bullets	The vector of bullets belongs to Player since the player creates the bullets. The bullets are kept in a Vector list so they can be iterated over and updated in Main. Since they need to be accessed in Main, the variable is public. We decided to do a Vector over an array since we’re unsure of how many bullets will be on screen at a time, so an array wouldn’t do, and Vectors

	have less ConcurrentModification Errors than ArrayLists.
Main game	We access Main's height and width to determine where to set the player once the game starts.
Sphere s	Sphere s is called in a method that detects what to do when player is in contact with a sphere (take damage.)

3. What the constructor(s)?

```
public Player(){
  bullets = new Vector<Bullet>();
  health = 100;
  healthString = ""+health+"";
  centerX = game.width/2 + radius;
  centerY = game.height - radius;
  radius = 50;
  velocityX = 0;
}
```

If there is only one, why? There's only one type of Player that exists in our game, and that player will always have these variables set this way when the game is created, so there's no need for another constructor.

4. What are the methods (return type, name, argument types)?

What do they do (in words, not code)?

Why are they public/private/static/not-static?

Name and Type	Notes
public void update( )	This method makes the Player in charge of updating itself. The method adds the velocity to centerX (since Player can only move horizontally) and also makes sure that the player cannot move itself offscreen. This method is public because it'll be used in Main's update method.
public void draw (Graphics g)	This method makes the Player in charge of drawing itself. It sets itself to be blue and then draws itself at the a certain point in relation to centerX and centerY, and its width and height are twice the radius (the diameter). It also draws the Player's health on its body. This method is public because it'll be used in Main's paintComponent method.
public boolean hitDetection(Sphere s)	This method returns whether or not Player is in contact with a sphere. It finds the result from the distance formula with the coordinates of the centers of both a sphere and Player and compares it to the sum of their radii to determine this. This method is used in tandem with decreaseHealth in Main, so needs to be public.
public void decreaseHealth (boolean hit)	The purpose of this method is to decrease the health of Player steadily while it's in contact with a sphere. It takes in a boolean to determine if the Player is being hit with a sphere. This boolean is provided from hitDetection, as those methods are used in tandem in the collide method in Main. If the player is in contact with a sphere, their health will go down by one in every frame of contact. This method is used in Main, so it's public.

public void shoot ()	This method adds a new bullet to the Vector list of bullets. In order for the bullets to function, they need to be updated and drawn, which Main is in charge of, so the method needs to be public.
public boolean isDead( )	This method checks to see if the Player has more than 0 health. If the player does have 0 or less health, boolean dead will be set to true and the player cannot move anymore. This method will be used in Main to see if the game should be ended or not, so this method needs to be public.

Class: Sphere

1.Describe the semantics and use of the class. What does it represent? When your program is run, does one instance exist? or a few? or many?

The Sphere class represents the enemies of the game: the spheres. They will enter the screen and bounce off of themselves and the walls in hopes to damage the player. Once the game is run, spheres will be generated randomly in various positions across the screen. This number should increase as the player gets farther along in the game.

2. What are the member variables (names and types)? What does each represent semantically?Why are they public/private?Why are they the type they are?

Name and Type	Notes
public double velocityX	VelocityX is meant to keep track of each sphere's velocity in the horizontal direction. This variable is public because it will be utilized in game's physics. VelocityX is a double because velocities are represented numerically, and we chose double over int to be more precise with the numbers.
public double velocityY	VelocityY is meant to keep track of each sphere's velocity in the vertical direction. This variable is public because it will be utilized in game's physics. VelocityY is a double because velocities are represented numerically, and we chose double over int to be more precise with the numbers.
public double centerX	CenterX keeps track of the x coordinate of the center of each sphere on the screen. This value is meant to be used in tandem with centerY. This variable is public in order to be used in Main's collide method. This variable is a double because it will be used in methods that will bounce the spheres off the walls and off of each other, and to get the most precise bounces, a double's level of precision is better than an int's.
public double centerY	CenterY keeps track of the y coordinate of the center of each sphere on the screen. This value is meant to be used in tandem with centerX. This variable is public in order to be used in Main's collide method. This variable is a double because it will be used in methods that will bounce the spheres off the walls and off of each other, and to get the most precise bounces, a double's level of precision is better than an int's.
public double radius	Radius is meant to keep track of the distance between the center of each sphere to any edge of the circle. This value is public as it will be used in the collide method in Main. We chose this variable to be a double for precision in equations.
public final int initHealth	InitHealth is meant to keep track of every sphere's starting health. This is public since every sphere's initial health will be added to the overall score once it is killed. This variable is an int because health is easy to represent numerically, but it does not need precision like a double. This variable is final because it should not be tampered with or changed.
public int health	Health keeps track of a sphere's health over the course of the game. Health will be depleted while the sphere is shot while initHealth is a static value that will not be changed. It's easy to represent health numerically, but we don't need decimals, so we chose an int to represent health.
public String healthDisplay	HealthDisplay is a string that will display the health of each sphere on the sphere as the game goes on. HealthDisplay is a string in order to be used by the drawString method in the Graphics api. HealthDisplay gets updated alongside health.

public Color color	Color is represents the color of each sphere, which will be random.
Random rand	We need random to randomize the size, speeds, colors, and health of the spheres once created.
Main game	Main is a variable in order to be referenced in the bounce method.

3. What the constructor(s)?

```
public Sphere(int k, int j){
    velocityX = rand.nextInt(4)-2;
    velocityY = rand.nextInt(3)+2;
    centerX = k;
    centerY = j;
    radius = rand.nextInt(10)+29;
    initHealth = rand.nextInt(95)+5;
    health = initHealth;
    healthDisplay = " "+ health +" ";
    color = new Color(rand.nextFloat(), rand.nextFloat(), rand.nextFloat());
}
```

4. What are the methods (return type, name, argument types)?

What do they do (in words, not code)?

Why are they public/private/static/not-static?

Name and Type	Notes
public void decreaseHealth (int damage)	The purpose of this method is to decrease the health of the sphere once shot by a bullet. All it does it decrease the health by the numerical value of the damage from the bullet. This method is public because it's utilized in the collide method in Main.
public void switchSignsX( )	SwitchSignsX just flips the velocityX value of the sphere. This method is public because it's used in the collide method in Main once the spheres collide with each other.
public void switchSignsY( )	SwitchSignsY does the same thing semantically for velocityY.
public void update( )	This method makes every sphere in charge of updating itself. The respective velocities are added to their respective centers (centerX + velocityX, centerY + velocityY), the healthDisplay is constantly updated to continually display the health of the sphere, and bounce off of the walls in Main. This method is public because it'll be used in Main's update method.
public void draw (Graphics g)	This method makes every sphere in charge of drawing itself. It sets itself to be a random color and then draws itself at the a certain point in relation to centerX and centerY, and its width and height are twice the radius (the diameter). It also draws the healthDisplay on the sphere. This method is public because it'll be used in Main's paintComponent method.
private void bounce (Main game)	This method makes sure that the sphere bounces off the walls in Main. Basically, if the left or right edge of the sphere starts to go off the left or right side of the screen, flip velocityX and reposition the

	sphere so they're not caught in the wall. If the the top or bottom edge of the sphere starts to go off the top or bottom of the screen, flip velocityY and reposition. This method is private because it's only called in the sphere's update method, which will account for bouncing in Main.

Class: Bullet

1.Describe the semantics and use of the class. What does it represent? When your program is run, does one instance exist? or a few? or many?

The Bullet class is an extension of the sphere class. The bullets represent the player's option of defeating the spheres that plague them. By holding the space key or W, the player will shoot an endless stream of bullets that don't do too much damage on their own, but in succession pack a punch. They will enter the screen above the player and travel upward. If they collide with a sphere, it'll deal damage to it. Once the game is run, player will create many instances of Bullet.

2. What are the member variables (names and types)? What does each represent semantically?Why are they public/private?Why are they the type they are?

Since Bullet extends Sphere, a lot of the member variables function the same way as in sphere. If you'd like a detailed description, read Sphere's member variables' descriptions.

Name and Type	Notes
public double velocityX	See Sphere Since the bullets will never travel horizontally, velocityX is not used for bullets.
public double velocityY	See Sphere The bullets travel solely upward at a constant rate.
public double centerX	See Sphere
public double centerY	See Sphere
public double radius	See Sphere
public final int initHealth	See Sphere
public int health	See Sphere We gave the spheres all a health of 1 so when they collide with a sphere, it'll "die", which means it won't be drawn anymore.
public String healthDisplay	See Sphere We didn't want each bullet to have a tiny 1 on it, so we set healthDisplay to be nothing. The player should not be concerned with the health of their bullets.
public Color color	See Sphere

3. What the constructor(s)? public Sphere(int k, int j)

If there is only one, why? Bullet only needs one constructor because this is the only way bullets need to be called. K represents the x coordinate of the center and j represents the y coordinate.

4. What are the methods (return type, name, argument types)?

What do they do (in words, not code)?

Why are they public/private/static/not-static?

Again, Bullet extends Sphere, so a lot of these methods will have the same properties.

Name and Type	Notes
public void decreaseHealth (int damage)	See Sphere. In Bullet, this method was mainly used to make sure the bullet's health reached 0 so it could be erased from the screen once "dead".
public void switchSignsX( )	See Sphere This method is not used for Bullet. Not only is there no horizontal velocity, but there's no need to flip it either.
public void switchSignsY( )	See Sphere. This method is not used for Bullet. The bullet's direction will not be changed.
public void update( )	See Sphere The only difference between Sphere's update and Bullet's update is the fact that bullets do not display their own health, so there's no need to update it.
public void draw (Graphics g)	See Sphere.
public void bounce (Main game)	See Sphere. The bullets don't bounce of the edge of world. Once they travel off the screen, they're removed from the Vector list.

Interface Interactable

Interactable is an interface that was sets if a class is going to be able to be updated and drawn or not. While it does seem barren, we had originally planned to have more classes that implemented Interactable, like powerups, but we ran out of time. The methods draw and update have been explained.