SOFTWARE METHOD AND TOOLS
(CS-5540)

ASSIGNMENT 6

-RASHMI TRIPATHI
(rtrh6@mail.umkc.edu)

Project Level Changes

As part of this project I have done following changes in code to create my test cases:

- I have changed all the private and protected variables to public.
- I have added one parametrized constructor in Player class. Here it will allow us to add player at any position we want to on the AWT screen.

```
public Player(Vector2 position) {
super(position, new Vector2(0.0, 0.0), 10.0, 0);
this.bullets = new ArrayList<>();
this.rotation = DEFAULT_ROTATION;
this.thrustPressed = false;
this.rotateLeftPressed = false;
this.rotateRightPressed = false;
this.firePressed = false;
this.firingEnabled = true;
this.fireCooldown = 0;
this.overheatCooldown = 0;
this.animationFrame = 0;
}
```

- I have added one parametrized constructor in Asteroid class. Here it will allow us to add asteroid at any position we want to on the AWT screen.

```
public Asteroid(Vector2 position,Vector2 velocity) {
              super(position,velocity, AsteroidSize.Large.radius,
                    AsteroidSize.Large.killValue);
                  this.size = AsteroidSize.Large;
      }
```

- I have also added initGame() method in Game as calling startGame() method will start AWT screen and then test cases will not be executed.

```
 public void initGame()
{
 this.random = new Random();
 this.entities = new LinkedList<Entity>();
 this.pendingEntities = new ArrayList<>();
 this.player = new Player();

 //Set the variables to their default values.
 resetGame();

 //Create the logic timer and enter the game loop.
 this.logicTimer = new Clock(FRAMES_PER_SECOND);

 }
```

Assumptions:

The player and asteroid should not move at high speed as it seems to be.

Score should be updated if bullet and asteroid get hit accordingly.

Any collision between asteroid and player will result in life loss

---

1. **What problems did you find in the code? For each problem, further explain what test case you used to find the problem.**

   I have created 7 test cases and out of which three test cases are failing. These are described below:

S.no	Class -> Method Tested	Test Case Description	Result
1	Game -> updateGame	check for total entities as start of game	Pass
2	Game -> updateGame	Check for lives after collsion between asteroid and player and then score update.	Pass for lives, Fails for score
3	Game -> updateGame	Check for score update after collision between asteroid and bullet	Pass
4	Game -> updateGame	Check for game level update	Pass
5	Entity -> update	Check for new positions after asteroid movement	Fail
6	Entity -> update	Check for new positions after player movement	Fail
7	Entity -> checkCollision	Check for collision between asteroid an player	Pass

I have explained the Game -> updateGame() specifically in next section. Here are the bugs found in the code as per my test cases logic:

Failures Scenario:

1) **The score get updated when asteroid and player collides. I have found it through test case 2.**

   As part of this I have added Player at one position and asteroid at the same position as player. Then I have called updateGame method of game which will check for collision and then lives got decremented.

However the score should not have updated by 20 . The application added the large asteroid kill value to total score which is not correct.

Test Case Description	Expected Output	Actual Output	Result
Check for collision between player & asteroid	Lives =2	Lives=2	Pass
	Score=0	Score=20	fail

Following is the code for the same:

```
Double oldScore= (double)asteroidGame.getScore();
Entity asteroid=new Asteroid(new Random());
asteroidGame.registerEntity(asteroid);
asteroidGame.player=new Player(asteroid.getPosition());
asteroidGame.registerEntity(asteroidGame.player);

asteroidGame.player.update(asteroidGame);

//before collision i should be three
assertEquals(3, asteroidGame.lives);

asteroidGame.updateGame();
assertEquals(2, asteroidGame.lives);

Double newScore=(double)asteroidGame.getScore();

assertEquals(oldScore, newScore);
```

2) Asteroid is moving at high speed than expected. This is tested by Test case 5.

As part of this I have added asteroid , got its old velocity and position. After that I updated the game.

The new positions expectation are

New position = old position +velocity;

However for X coordinate , it will getting 3 point up and on Y axis it is 1 point up. Because of which asteroids are moving with very high speed before the player can even fire.

Test Case Description	Expected Output	Actual Output	Result
Check for asteroid new position	Position.x= 116.75 Position.y= 278.42	Position.x= 428.72 Position.y= 278.42	Fail

Following is the code for the same:

```
Entity asteroid=new Asteroid(new Random());
asteroidGame.registerEntity(asteroid);

Vector2 asteroidOldPos=asteroid.getPosition();
Double oldX=asteroidOldPos.x;
Double oldY=asteroidOldPos.y;

Double oldVelocityX=asteroid.getVelocity().x;

asteroid.update(asteroidGame);
asteroidGame.updateGame();

Vector2 asteroidNewPos=asteroid.getPosition();
Double newX=asteroidNewPos.x;
Double newY=asteroidNewPos.y;

Double velocityXAfterThrust=asteroid.getVelocity().x;
Double velocityYAfterThrust=asteroid.getVelocity().y;

Double asteroidNewExpectedPosforX=oldX+velocityXAfterThrust;
Double asteroidNewExpectedPosforY=oldY+velocityYAfterThrust;


//old and new positions of the player after thrust should not be same
assertNotEquals(oldX,newX);
assertNotEquals(oldY,newY);

//the new position of the player should move by just one notch of velocity
assertEquals(asteroidNewExpectedPosforX,newX);
assertEquals(asteroidNewExpectedPosforY,newY);
```

3) **When player thrust is enabled, the player moves by 3 times as velocity is multiplied by 3 while adding it to position. This is tested by test case 6.**

As part of this I have thrusted the player and then called update game. After this the old and new positions should not be same.

Also the new position expected coordinated should move by one notch of velocity which did not happen.

Test Case Description	Expected Output	Actual Output	Result
Check for player new position after thrust and rotation	Position.x= 274.98 Position.y= 274.96	Position.x= 274.92 Position.y= 274.96	Fail

Following is the code for the same:

```
Entity asteroid=new Asteroid(new Random());
asteroidGame.registerEntity(asteroid);

Vector2 playerOldPos=asteroidGame.player.getPosition();
Double oldX=playerOldPos.x;
Double oldY=playerOldPos.y;

Double oldVelocityX=asteroidGame.player.getVelocity().x;

asteroidGame.player.setThrusting(true);
asteroidGame.player.setRotateLeft(true);
asteroidGame.player.update(asteroidGame);
asteroidGame.updateGame();

Vector2 playerNewPos=asteroidGame.player.getPosition();
Double newX=playerNewPos.x;
Double newY=playerNewPos.y;

Double velocityXAfterThrust=asteroidGame.player.getVelocity().x;
Double velocityYAfterThrust=asteroidGame.player.getVelocity().y;

Double playerNewExpectedPosforX=oldX+velocityXAfterThrust;
Double playerNewExpectedPosforY=oldY+velocityYAfterThrust;


//old and new positions of the player after thrust should not be same
assertNotEquals(oldX,newX);
assertNotEquals(oldY,newY);

//the new position of the player should move by just one notch up
assertEquals(playerNewExpectedPosforX,newX);
assertEquals(playerNewExpectedPosforY,newY);
```

Success Scenario:

1) Collision between bullet and large asteroid yielding correct score

As part of this I have created asteroid at same position as player on X axis and then fired bullets. Then called updateGame method . It checked for collision and updated score.

Test Case Description	Expected Output	Actual Output	Result
Check for collision between bullet and asteroid	Score=40	Score=40	Pass

Following is the code for the same:

```
int oldScore=asteroidGame.getScore();

 Asteroid asteroid=new Asteroid(new
 Vector2(player.getPosition().x,100),player.getVelocity());
 asteroidGame.registerEntity(asteroid);
 asteroidGame.updateGame();

 Double expectedScore=(double)oldScore+asteroid.getKillScore();

 //start firing
 for(int i=0;i<20;i++)
 {
        Thread.sleep(100);
        Entity bullet=new Bullet(player, player.getRotation());
        player.update(asteroidGame);
        asteroidGame.registerEntity(bullet);
        asteroidGame.updateGame();

 }

 Double currentScore=(double)asteroidGame.getScore();

 assertEquals(expectedScore,currentScore);
```

2) **Test collision check flag between asteroid and player**

As part of this I have added player an asteroid at different position and checked for collision. It did not happen.
Also I have added player and asteroid at same position. The checkCollision method returned true.

Test Case Description	Expected Output	Actual Output	Result
Check for collison between player and asteroid **Player and Asteroid at same position**	True	True	Pass
Player and Asteroid at differnt position	false	false	Pass

Following is the code for the same:

```
Entity asteroid=new Asteroid(new Random());
```

```
asteroidGame.registerEntity(asteroid);

Vector2 asteroidPos=asteroid.getPosition();
 Vector2 playerPos=asteroidGame.player.getPosition();

//as positions are different first time, collision method should return
false
assertNotEquals(asteroidPos.x, playerPos.x);
assertNotEquals(asteroidPos.y, playerPos.y);
assertFalse(asteroidGame.player.checkCollision(asteroid));

asteroidGame.player=new Player(asteroid.getPosition());
asteroidGame.registerEntity(asteroidGame.player);
asteroidGame.player.update(asteroidGame);
asteroidGame.updateGame();

asteroidPos=asteroid.getPosition();
 playerPos=asteroidGame.player.getPosition();

assertTrue(asteroidGame.player.checkCollision(asteroid));
```

3) Test Game level after all enemies are dead

As part of this I have killed all asteroids and then checked the game level. It should go one notch up.

Test Case Description	Expected Output	Actual Output	Result
Check game level after all asteroids/enemies are dead	Level=1	Level=1	Pass

Following is the code for the same:

```
//remove existing asteroid
removeAsteroids();
//first update to remove dead entities
asteroidGame.updateGame();
//second call to make level up
asteroidGame.updateGame();

assertEquals(1, asteroidGame.getLevel());
```

4) Checking for Game entities.

As part of this I have added five asteroids and updated game. Then checked for existing entities and pending entities.

Test Case Description	Expected Output	Actual Output	Result
Check for total entities at start of the game	Entities=6 PendingEntities=0	Entities=6 PendingEntities=0	Pass

Following is the code for the same:

```
for(int i=0;i<5;i++)
{
	Asteroid a=new Asteroid(new Random());
	asteroidGame.registerEntity(a);
}

asteroidGame.updateGame();

assertEquals(0, asteroidGame.pendingEntities.size());
//five asteroids we have added  and one player
assertEquals(6, asteroidGame.entities.size());
```

---

2. Specifically explain the test case that you have created for the updateGame method of Class Game. What is your input, and what is your expected output? What is your logic of testing this method?

The following functions are performed by updateGame method:

- Setting and resetting entities list and pending entities list
- Handle collisions and in reverse update score
- Check for collisions
- Making level up in case all enemies are dead
- Checking for is game over or not and taking corresponding actions.

I have created the following four test case for the same:

Test Case	Input	Actual Output	Expected Output	Result
Game Entities	Existing player Asteroids	game.entities=6 game.pendingEntities=0	game.entities=6 game.pendingEntities=0	Pass
Game Collision	Asteroid, Player	Game.lives =2	Game.lives=2	Pass
		Game.score=20	Game.score=0	Failed

Game Score	Bullet, asteroid, player	Game.score=40	Game.score=40	Pass
Game Level	Killed all asteroids	Game.level=1	Game.level=1	Pass

The following process was followed to test each case:

- `testUpdateGameEntities`

  Added five asteroids on my own and then checking for all entities and pending entities. As updateGame() clear the pendingEntities and add all the pendingEntities to entities, the same get updated.

```
assertEquals(0, asteroidGame.pendingEntities.size());
assertEquals(6, asteroidGame.entities.size());
```

- `testUpdateGameCollision`

  Created one asteroid at random position and added player at the same position as asteroid. As the distance between them is zero, update method will kill player hence decrementing the number of lives.
  Also as player and asteroid collision should have resulted in player life loss and no score update.
  Hence this is a bug that when a player and asteroid collide, the score get updated.
  This is tested by this usecase.

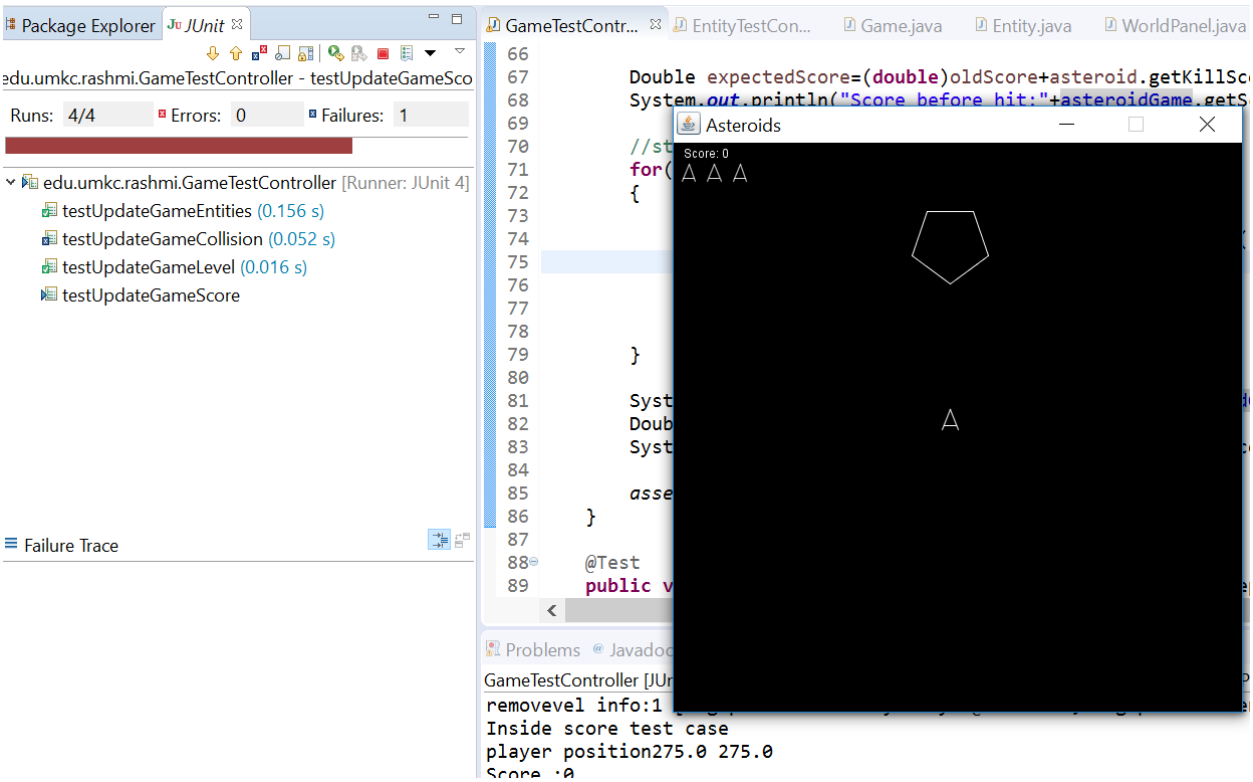
```
assertEquals(2, asteroidGame.lives);
assertEquals(oldScore, newScore);
```

- `testUpdateGameScore`

  Added one large asteroid. The position for the same is decided by player position.

  Asteroid X axis position is same as player and Y position is given by me. Now we have created bullets.

  Below is the screenshot for the same:



After update the bullets should hit the asteroid and the score should get updated to killed asteroid kill value.

```
Double expectedScore=(double)oldScore+asteroid.getKillScore();
//after kill
assertEquals(expectedScore,currentScore);
```

- `testUpdateGameLevel`

  As part of this testcase , we have first killed all the asteroids and then checked whether the level got up or not.

  To kill asteroids we need to make property of entity i.e needsRemoval as true.
  Update game method will remove all dead entities and on updateGame method call again as there as no enemies the level should go up.

```
//remove existing asteroid
removeAsteroids();
assertEquals(1, asteroidGame.getLevel());
```

---

3. Include a screenshot of the result of running your test suite.

Screenshot is as follows: