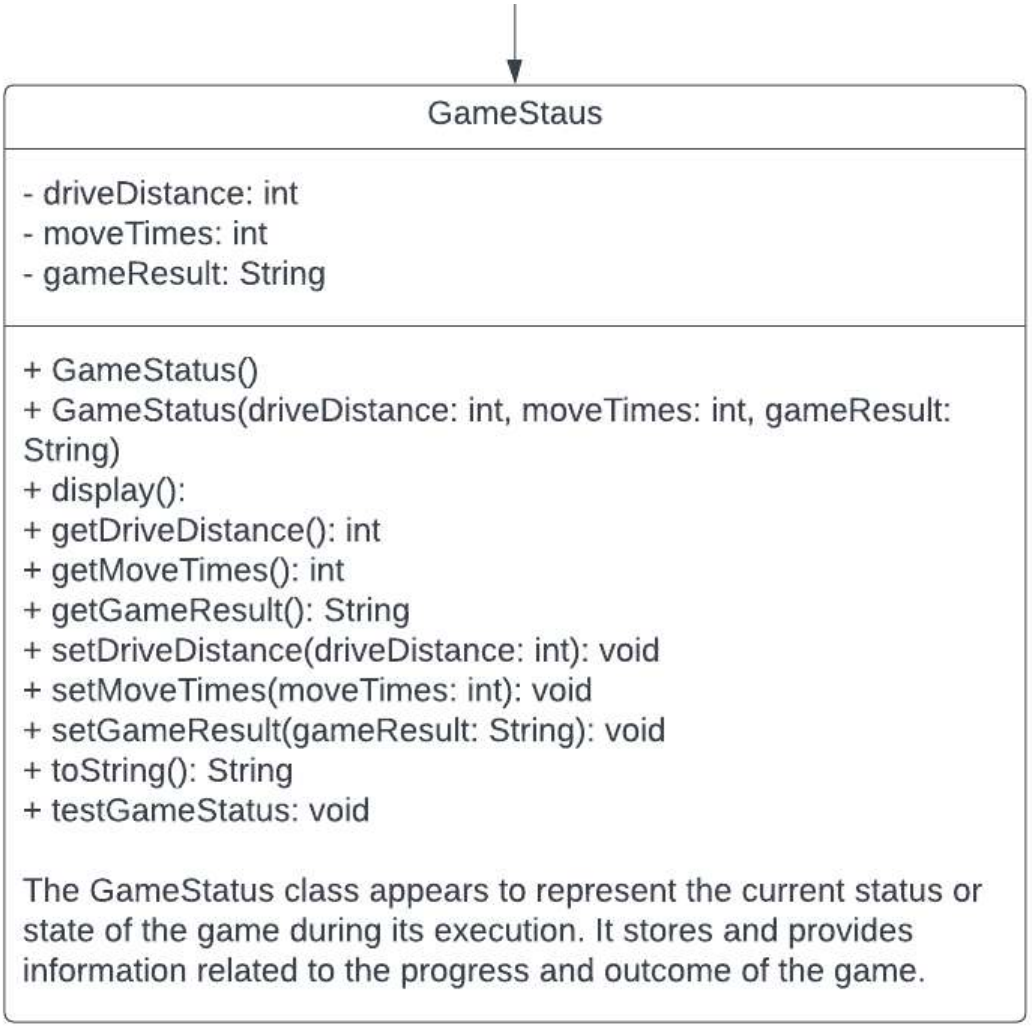
TestStrategy Version 13 Author ZiqiPei

1.GameStatus class test



1. Test Default Constructor

Test Plan for GameStatus Class

Test Default Constructor

1. Create a `GameStatus` object using the default constructor.
2. Create a `GameStatus` object using the non-default constructor with invalid field valu

Test Non-Default Constructor with Valid Field Values

Test Non-Default Constructor with Invalid Field Values

Unit Test

3Test all get methods:

Test getDriveDistance()

Test getMoveTimes()

Test getGameResult()

4Test all set method

Test setDriveDistance()
with valid Field Values

with an invalid value (e.g., -10).

Test setMoveTimes()
with Valid Field Values

with Invalid Field Values

Test setGameResult()
with Valid Field Values

with a Invalid value.

- **Test Steps:**
    1. Create a `GameStatus` object using the default constructor.
    2. Display the information using the `display` method.

- **Expected Output:**

```
Drive Distance: 1
Move Times: 1
Game Result: " "
```

- Actual **Output:**

```
Create an GameStatus object with the default constantor
Drive Distance :1
Move Times :1
Game Result :
```

2. Test Non-Default Constructor with Valid Field Values

- **Test Steps:**
  1. Create a `GameStatus` object using the non-default constructor with valid field values (e.g., driveDistance: 75, moveTimes: 30, gameResult: "Game Over").
  2. Display the information using the `display` method.
- **Expected Output:**

```
Drive Distance: 75
Move Times: 30
Game Result: "Game Over"
```

  - Actual **Output:**

```
Create an GameStatus object with the non-default constantor with valid  filed values
Drive Distance :75
Move Times :30
Game Result :GameOver
```

3. Test Non-Default Constructor with Invalid Field Values

- **Test Steps:**

1. Create a `GameStatus` object using the non-default constructor with invalid field values (e.g., driveDistance: -10, moveTimes: -5, gameResult: "12122").
2. Display the information using the `display` method.

- **Expected Output:**

```
Drive Distance: 1
Move Times: 1
Game Result: " "
```

- Actual **Output:**Test FAILED!

```
Create an GameStatus object with the non-default constantor with invalid filed value
Drive Distance :-10
Move Times :-5
Game Result :Invalid
```

Method not validating length of staffId and value of payScale correctly.
Modify this line

```
 */
public GameStatus(int driveDistance, int moveTimes, String gameResult)
{
    Validation validator = new Validation();// Create an instance of Validation getClass

    // Validate driveDistance
    if(driveDistance <= 0)
    {
        //If invalid ,set default value.
        this.driveDistance = 1;// Default value for invalid drive distance.
    } else
    {
        this.driveDistance = driveDistance;
    }
    // Validate moveTimes
    if(moveTimes <= 0)
    {
        //If invalid ,set default value;
        this.moveTimes = 1;
    }
    else
    {
        this.moveTimes = moveTimes;
    }
    // Set gameResult directly , assumeing no
    if(validator.isBlank(gameResult) || !("message".equals(gameResult)))
    {
        this.gameResult = " ";
    }else
    {
        this.gameResult = gameResult;
    }
}
```

```
Create an GameStatus object with the non-default constantor with invalid filed value
Validator initialized
Drive Distance :1
Move Times :1
Game Result :
```

Test 4.1: Test `getDriveDistance` Method

Test Steps:

1. Create a `GameStatus` object.
2. Set driveDistance to 100.
3. Get driveDistance using `getDriveDistance`.

Expected Output:

```
Expected Drive Distance: 100
Actual Drive Distance: 100
```

```
System.out.println("Test Get Dirve Distance");
GameStatus gameStatus4 = new GameStatus();
int expectedDriveDistance = 100;
gameStatus4.setDriveDistance(expectedDriveDistance
int actualDriveDistance = gameStatus4.getDriveDist
if(expectedDriveDistance == actualDriveDistance)
{
    System.out.println("Test passed:");
}
else
{
    System.out.println("Test failed:");
}
```

Actual **Output:**

```
Test Get Dirve Distance
Test passed:
```

Test 4.2: Test `getMoveTimes` Method

- **Test Steps:**
  1. Create a `GameStatus` object.
  2. Set moveTimes to 50.
  3. Get moveTimes using `getMoveTimes`.
- **Expected Output:**

```
Expected Move Times: 50
Actual Move Times: 50
```

```
System.out.println("Test getMoveTimes");
GameStatus gameStatus5 = new GameStatus();
int expectedMoveTimes1 = 50;
gameStatus5.setMoveTimes(expectedMoveTimes1);
int actualMoveTimes1 = gameStatus5.getMoveTimes(
if(expectedMoveTimes1 == actualMoveTimes1)
{
    System.out.println("Test passed:");
}
else
{
    System.out.println("Test failed:");
}
```

Actual **Output:**

```
Test getMoveTimes
Test passed:
```

Test 4.3: Test `getGameResult` Method

- **Test Steps:**
  1. Create a `GameStatus` object.
  2. Set gameResult to "Game Over".
  3. Get gameResult using `getGameResult`.
- **Expected Output:**

```
Expected Game Result: "Game Over"
Actual Game Result: "Game Over"
```

```
System.out.println("Test getGameResult");
GameStatus gameStatus6 = new GameStatus();
String expectedGameResult = "Game Over";
gameStatus6.setGameResult(expectedGameResult);
String actualGameResult = gameStatus6.getGameResult
if(actualGameResult == expectedGameResult)
{
    System.out.println("Test passed:");
}
else
{
    System.out.println("Test failed:");
}
```

Actual **Output:**

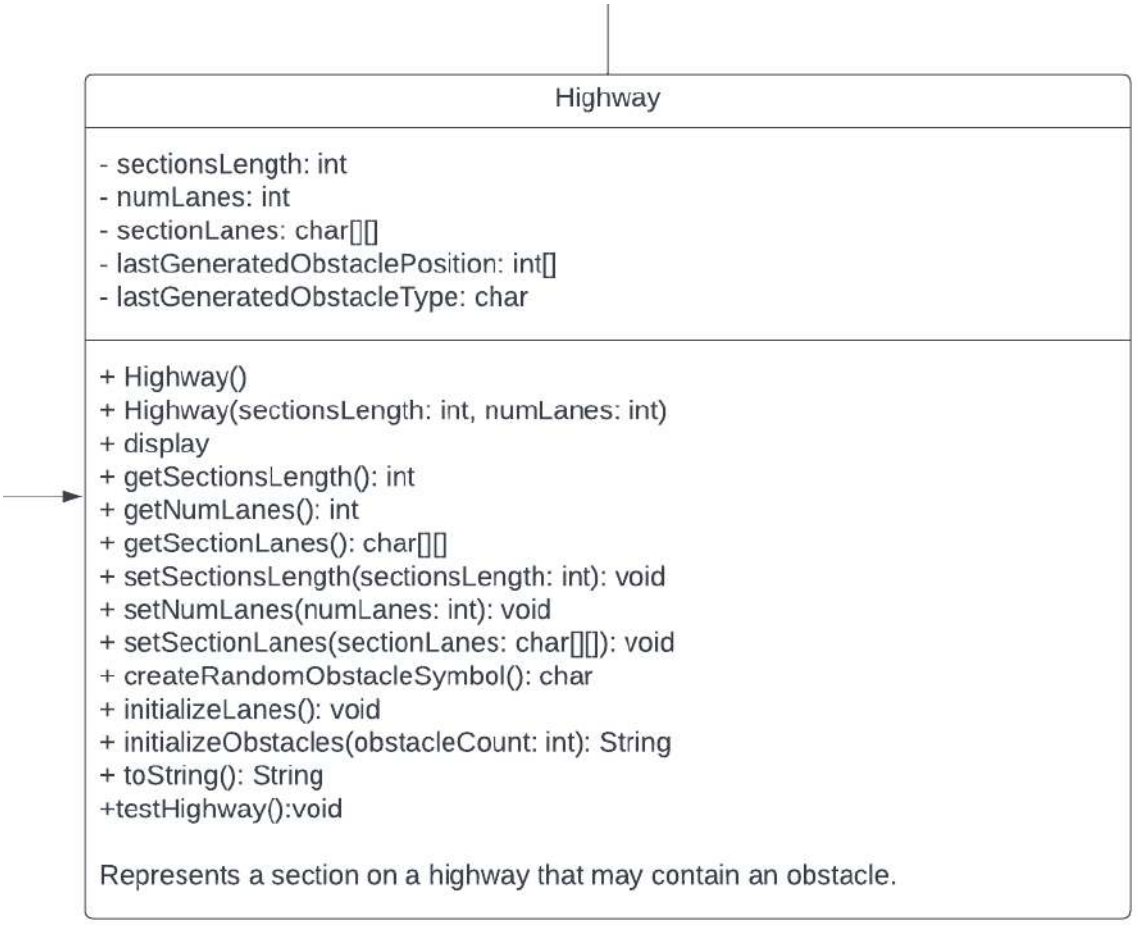
```
Test getGameResult
Test passed:
```

```
public void testGameStatus()
    {
        System.out.println("Create an GameStatus object with
        GameStatus gameStatus1 = new GameStatus();
```

```
        gameStatus1.display();

        System.out.println("Create an GameStatus object with
        GameStatus gameStatus2 = new GameStatus(75, 30, "Game
        gameStatus2.display();

        System.out.println("Create an GameStatus object with
        GameStatus gameStatus3 = new GameStatus(-10, -5, "121
        gameStatus3.display();

        System.out.println("Test Get Dirve Distance");
        GameStatus gameStatus4 = new GameStatus();
        int expectedDriveDistance = 100;
        gameStatus4.setDriveDistance(expectedDriveDistance);
        int actualDriveDistance = gameStatus4.getDriveDistanc
        if(expectedDriveDistance == actualDriveDistance)
        {
            System.out.println("Test passed:");
        }
        else
        {
            System.out.println("Test failed:");
        }

        System.out.println("Test getMoveTimes");
        GameStatus gameStatus5 = new GameStatus();
        int expectedMoveTimes1 = 50;
        gameStatus5.setMoveTimes(expectedMoveTimes1);
        int actualMoveTimes1 = gameStatus5.getMoveTimes();
        if(expectedMoveTimes1 == actualMoveTimes1)
        {
            System.out.println("Test passed:");
        }
        else
        {
            System.out.println("Test failed:");
        }
```

```
        System.out.println("Test getGameResult");
        GameStatus gameStatus6 = new GameStatus();
        String expectedGameResult = "Game Over";
        gameStatus6.setGameResult(expectedGameResult);
        String actualGameResult = gameStatus6.getGameResult()
        if(actualGameResult == expectedGameResult)
        {
            System.out.println("Test passed:");
        }
        else
        {
            System.out.println("Test failed:");
        }
    }
```

2Highway test



Test Plan for Highway Class

1. **Test Default Constructor**
   - Create a `Highway` object using the default constructor.
   - Display information about the highway.
2. **Test Non-Default Constructor with Valid Field Values**
   - Create a `Highway` object using the non-default constructor with valid field values.
   - Display information about the highway.
3. **Test Non-Default Constructor with Invalid Field Values**
   - Create a `Highway` object using the non-default constructor with invalid field values.
   - Display information about the highway.
4. **Unit Tests**

- **Test getSectionsLength()**
  - Get and display the sections length of the highway.
- **Test getNumLanes()**
  - Get and display the number of lanes in the highway.
- **Test getSectionLanes()**
  - Get and display the layout of highway lanes.
- **Test setSectionsLength()**
  - Set the sections length with a valid value.
  - Display information about the highway.
  - Set the sections length with an invalid value (e.g., -5).
  - Display information about the highway.
- **Test setNumLanes()**
  - Set the number of lanes with a valid value.
  - Display information about the highway.
  - Set the number of lanes with an invalid value (e.g., 0).
  - Display information about the highway.
- **Test setSectionLanes()**
  - Set the layout of highway lanes.
  - Display information about the highway.
- **Test createRandomObstacleSymbol()**
  - Generate and display a random obstacle symbol.
- **Test initializeLanes()**
  - Initialize all highway lanes with empty spaces.
  - Display information about the highway.
- **Test initializeObstacles()**
  - Initialize obstacles on random lanes and positions in the highway.
  - Display information about the generated obstacles.

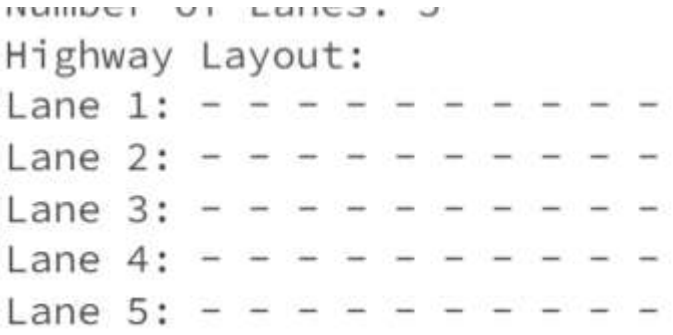
Actual Tests

Test 1: Default Constructor

- Object Initialization
- Expected Sections Length: 10
- Expected Number of Lanes: 5
- Expected Output:

```
Highway Information:
Sections Length: 10
Number of Lanes: 5
Highway Layout:
Lane 1:  ------------
Lane 2:  ------------
Lane 3:  ------------
Lane 4:  ------------
Lane 5:  ------------
```

Actual output

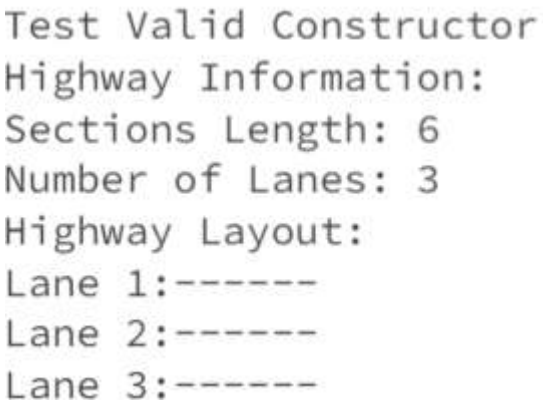


Test 2: Non-Default Constructor with Valid Field Values

- Test Data:
  Sections Length: 6
  Number of Lanes: 3
- Expected Output:

```
Highway Information:
Sections Length: 6
Number of Lanes: 3
Highway Layout:
Lane 1:------
Lane 2:------
Lane 3:------
```

Actual output



Test 3: Non-Default Constructor with Invalid Field Values

- Test Data:
  Sections Length: -2
  Number of Lanes: 0
- Expected Output:

```
Highway Information:
Sections Length: 10
Number of Lanes: 5
Highway Layout:
Lane 1:  ------------
Lane 2:  ------------
Lane 3:  ------------
Lane 4:  ------------
```

```
Lane 5:  ------------
```

Actual Output Test Failed

```
Create an Highway object withe the non-default constantor
Exception in thread "main" java.lang.NegativeArraySizeException: -2
        at Highway.<init>(Highway.java:37)
        at Highway.testHighway(Highway.java:217)
        at TestStrategy.main(TestStrategy.java:9)
```

Add this lane boolean function

```
 */
public Highway(int sectionsLength,int numLanes)
{
    //Validate parameters
    if(sectionsLength <= 0 || numLanes <= 0)
    {
        initializeLanes();
    }else
    {
        //Initialize the Highway
        this.sectionsLength = sectionsLength;
        this.numLanes = numLanes;
        this.sectionLanes = new char[numLanes][sectionsLe
    }
```

lastly output

```
Create an Highway object withe the non-default constanto
Highway Information:
Sections Length: 0
Number of Lanes: 0
Highway Layout:
```

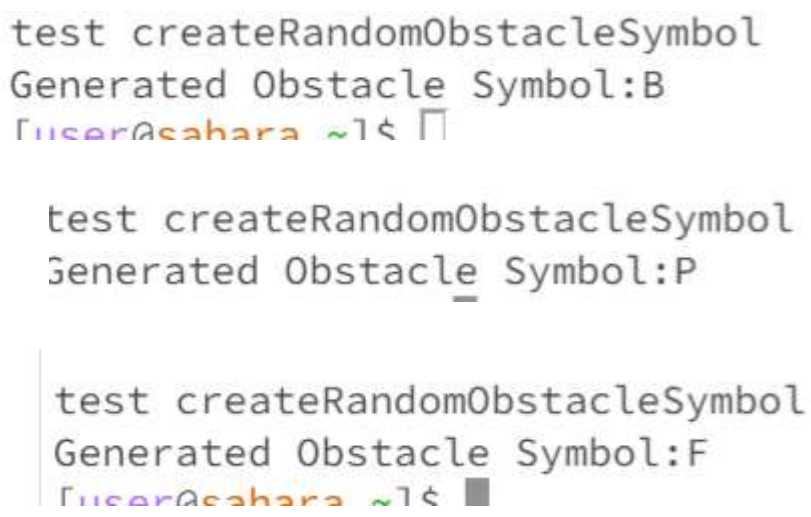
Test 4: Non-Default Constructor with Invalid Field Values

- Test Data:
  Sections Length: " ”
  Number of Lanes: 2
- Expected Output:
  return new Highway()
- Actual result

```
Create an Highway object with the non-default constantor
Validator initialized
Highway Information:
Sections Length: 32
Number of Lanes: 2
Highway Layout:
Lane 1:--------------------------------
Lane 2:--------------------------------
```

Test 5 TestRandomGenerator with Obstacle
Test Random crate Stmbol

Actual result



Test 6 Test Randomplace Obstacle
ObstacleCount :6
Expect Output: 6

Actual Output

```
Generate obstacle information:
 Generated obstacle at(2,8)of type:B
Generated obstacle at(1,3)of type:B
Generated obstacle at(2,6)of type:S
Generated obstacle at(2,7)of type:B
Generated obstacle at(3,4)of type:B
Generated obstacle at(3,7)of type:F

Updated Highway Information
Highway Information:
Sections Length: 10
Number of Lanes: 5
Highway Layout:
Lane 1: - - - - - - - - - -
Lane 2: - - -B- - - - - - -
Lane 3: - - - - - -S-B-B- -
Lane 4: - - - -B- - -F- - -
Lane 5: - - - - - - - - - -
```

Test7 TesttoString

Actual Output

```
Highway Information:
Highway{sectionLength=10, numLanes=5, sectionLength=[[C@78ac1102, [C@2de8284b, [C@396e2f
39, [C@a74868d, [C@12c8a2c0]}
```

Arrays.toString(sectionLanes) returns the output of the default toString method of a two-dimensional array. It will produce a result similar to [C@78ac1102, where [C represents a character array, and the following string of characters is the hash code of the array. .

```
public void testHighway()
    {
        System.out.println("Create an Highway object with the
        Highway highway1 = new Highway();
        highway1.display();

        System.out.println("Create an Highway object with the
```

```
        Highway highway2 = new Highway(6, 3);
        highway2.display();

        System.out.println("Create an Highway object with the
        Highway highway3 = new Highway(-1, -2);
        highway3.display();

        System.out.println("Create an Highway object with the
        Highway highway4 = new Highway(' ', 2);
        highway4.display();

        System.out.println("test createRandomObstacleSymbol")
        Highway highway5 = new Highway();
        char obstacleSymbol = highway5.createRandomObstacleSy
        System.out.println("Generated Obstacle Symbol:" + obs

        System.out.println("test set obstacle");
        Highway highway6 = new Highway();
        String obstacleInformation = highway6.initializeObsta
        System.out.println("Generate Obstacle Information:\n
        System.out.println("Updated Highway Information");
        highway6.display();

        System.out.println("Test toString");
        Highway highway7 = new Highway();
        String highwayInformation = highway7.toString();
        System.out.println("Highway Information:\n" + highway

    }
```

Test Plan for Player Class

3.1 Test Case: Default Constructor

Description

Verify that the `Player` class initializes with default values correctly.

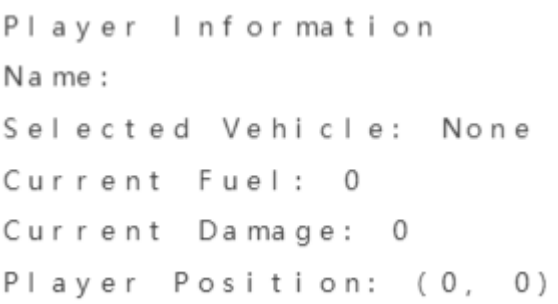
Steps

1. Create a `Player` object using the default constructor.
2. Display information about the player.

Expected Result

- Player Name: ""
- Selected Vehicle: Null
- Current Fuel: 0
- Current Damage: 0
- Player Position X: 0
- Player Position Y: 0

Actual Result test pass



3.2 Test Case: Non-Default Constructor with Valid Field Values

Description

Verify that the `Player` class initializes with valid field values correctly.

Steps

1. Create a `Player` object using the non-default constructor with valid field values.
2. Display information about the player.

Expected Result

- Player Name: "ziqi"
- Selected Vehicle: “Default”(new Vehicle())

- Current Fuel: 50 (or a valid initial fuel value)

```
// This checks if the selectedVehicle is not null it means a
        // useing the getType(),"None" This is a ternary oper
ja        System.out.println("Selected Vehicle: " + (selected
```

- Current Damage: 20 (or a valid initial damage value)
- Player Position X: 2
- Player Position Y: 0

```
Name: ziqi
Selected Vehicle: Default
Current Fuel: 100
Current Damage: 20
Player Position: (2, 0)
```

modify seter function

```
public void setSelectedVehicle(Vehicle selectedVehicle)
    {
        this.selectedVehicle = (selectedVehicle != null ? sel
    }
```

```
Player Information
Name: ziqi
Selected Vehicle: Vehicle{Type='Default', maxFuel=100, maxDamage=100}
Current Fuel: 100
Current Damage: 20
Player Position: (2, 0)
```

3.2 Test Case: Non-Default Constructor with InValid Field Values

Description

Verify that the `Player` class initializes with Invalid field values correctly.

Steps

1. Create a `Player` object using the non-default constructor with valid field values.
2. Display information about the player.

Expected Result

- Player Name: Default
- Selected Vehicle: null
- Current Fuel: 0
- Current Damage :0
- PlayerPosition(0,0)

test code

```
System.out.println("Create an Player object withe the non-defau
Player player3 = new Player();
player3.setPlayerName("Joohn");
player3.setSelectedVehicle(null);
player3.setCurrentFuel(-50);
player3.setCurrentDamage(-20);
player3.setPlayerPositionX(-2);
player3.setPlayerPositionY(0);
player3.display();
```

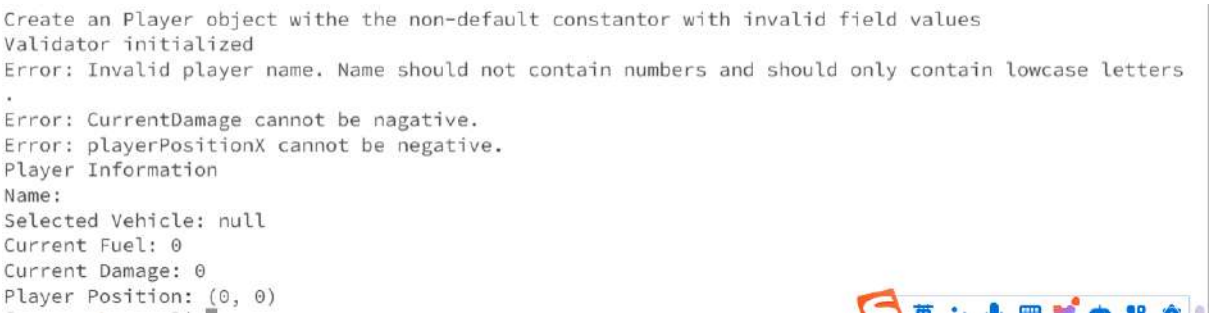
Actaul output

```
Player Position: (2, 0)
Create an Player object withe the non-default constantor with invalid field values
Player Information
Name: Joohn
Selected Vehicle: null
Current Fuel: 0
Current Damage: -20
Player Position: (-2, 0)
```

Change after show

```
Error: CurrentDamage cannot be nagative.
Error: playerPositionX cannot be negative.
Player Information
Name: Joohn
Selected Vehicle: null
Current Fuel: 0
Current Damage: 0
Player Position: (0, 0)
```

also need function to validation name invoke validation and found return




```
public void testPlayer()
    {
        System.out.println("Create an Player object with the
        Player player1 = new Player();
        player1.display();

        System.out.println("Create an Player object withe the
        Player player2 = new Player();
        player2.setPlayerName("ziqi");
        player2.setSelectedVehicle(new Vehicle());
        player2.setCurrentFuel(50);
        player2.setCurrentDamage(20);
        player2.setPlayerPositionX(2);
        player2.setPlayerPositionY(0);
        player2.display();

        System.out.println("Create an Player object withe the
```

```
        Player player3 = new Player();
        player3.setPlayerName("Joohn");
        player3.setSelectedVehicle(null);
        player3.setCurrentFuel(-50);
        player3.setCurrentDamage(-20);
        player3.setPlayerPositionX(-2);
        player3.setPlayerPositionY(0);
        player3.display();

    }
```

Test Plan for Player Class

4.1 Test Case: Default Constructor

Description

Verify that the DifficultyLevel class initializes with default values correctly.

Steps

1. Create a DifficultLevel object using the default constructor.
2. Display information about the player.

Expected Result

```
difficultyOption = " ";
minHighwayLength = 0;
maxHighwayLength = 0;
fuelCapacity = 0.0;
obstacleCount = 0;
currentHighwayLength = 0;
```

Actual Result test pass

```
Difficulty Option
Min Highway Length0
Max Highway Length0
Fuel Capacity0.0
Obstacle Count0
Current Highway Length0
```

3.2 Test Case: Non-Default Constructor with Valid Field Values

Description

Verify that the DifficultyLevel class initializes with valid field values correctly.

Steps

1. Create a DifficultLevel object using the non-default constructor with valid field values.
2. Display information about the player.

Expected Result

- Difficulty Option: Hard
- Min Highway Length : 31
- Max Highway Length : 50
- Fuel Capacity : 0.5
- Obstacle Count: 45
- Current Highway Length :0

```
Create an DifficultyLevel object with the non-defult constructor with valid field values
Difficulty Level Information
Difficulty Option: Hard
Min Highway Length: 31
Max Highway Length: 50
Fuel Capacity: 0.5
Obstacle Count: 45
Current Highway Length: 0
```

Test non-default constructor with invalid filed values(Test failed)
Create this function

```
Create an DifficultyLevel object with the non-default constantor with invalid field values
Difficulty Level Information
Difficulty Option: Invalid
Min Highway Length: -1
Max Highway Length: 50
Fuel Capacity: 1.5
Obstacle Count: -5
Current Highway Length: 0
```

```
Exception in thread "main" java.lang.IllegalArgumentException: Difficulty
tion cannot be null or empty.
        at DifficultyLevel.setDifficultyOption(DifficultyLevel.java:147)
        at DifficultyLevel.testDifficultyLevel(DifficultyLevel.java:295)
        at TestStrategy.main(TestStrategy.java:15)
```

```
public void setSelectedVehicle(Vehicle selectedVehicle)
    {
        this.selectedVehicle = (selectedVehicle != null ? sel
    }
```

```
Player Information
Name: ziqi
Selected Vehicle: Vehicle{Type='Default', maxFuel=100, maxDamage=100}
Current Fuel: 100
Current Damage: 20
Player Position: (2, 0)
```

3.2 Test Case: Non-Default Constructor with InValid Field Values

Description

Verify that the `Player` class initializes with Invalid field values correctly.

Steps

1. Create a `Player` object using the non-default constructor with valid field values.
2. Display information about the player.

Expected Result

- Player Name: Default
- Selected Vehicle: null

- Current Fuel: 0
- Current Damage :0
- PlayerPosition(0,0)

test code

```
System.out.println("Create an Player object withe the non-defau
Player player3 = new Player();
player3.setPlayerName("Joohn");
player3.setSelectedVehicle(null);
player3.setCurrentFuel(-50);
player3.setCurrentDamage(-20);
player3.setPlayerPositionX(-2);
player3.setPlayerPositionY(0);
player3.display();
```

Actaul output

```
Create an Player object withe the non-default constantor with invalid field values
Player Information
Name: Joohn
Selected Vehicle: null
Current Fuel: 0
Current Damage: -20
Player Position: (-2, 0)
```

Change after show

```
Error: CurrentDamage cannot be nagative.
Error: playerPositionX cannot be negative.
Player Information
Name: Joohn
Selected Vehicle: null
Current Fuel: 0
Current Damage: 0
Player Position: (0, 0)
```

also need function to validation name invoke validation and found return

Test setDifficultyLevel()
TestCode

```
public void setDifficultyLevel()
{

    System.out.println("Select difficulty");
    System.out.println("1. Easy");
    System.out.println("2. Moderate");
    System.out.println("3. Hard");
    System.out.println("Enter Your choice");
    Input input = new Input();
    int choice = input.acceptIntInput();
    switch(choice)
    {
        case 1:
            setDifficultyOption("Easy");
            setMinHighwayLength(10);
            setMaxHighwayLength(15);
            setFuelCapacity(1.0);
            setObstacleCount(12);
            break;
        case 2:
            setDifficultyOption("Moderate");
            setMinHighwayLength(16);
            setMaxHighwayLength(30);
            setFuelCapacity(0.8);
            setObstacleCount(24);
            break;
        case 3:
            setDifficultyOption("Hard");
            setMinHighwayLength(31);
            setMaxHighwayLength(50);
            setFuelCapacity(0.5);
            setObstacleCount(45);
            break;
        default:
            System.out.println("Invalid choice. Defaulting to Easy difficulty.");
            setDifficultyOption("Easy");
```

want output switch choice out put truely option

Actual output

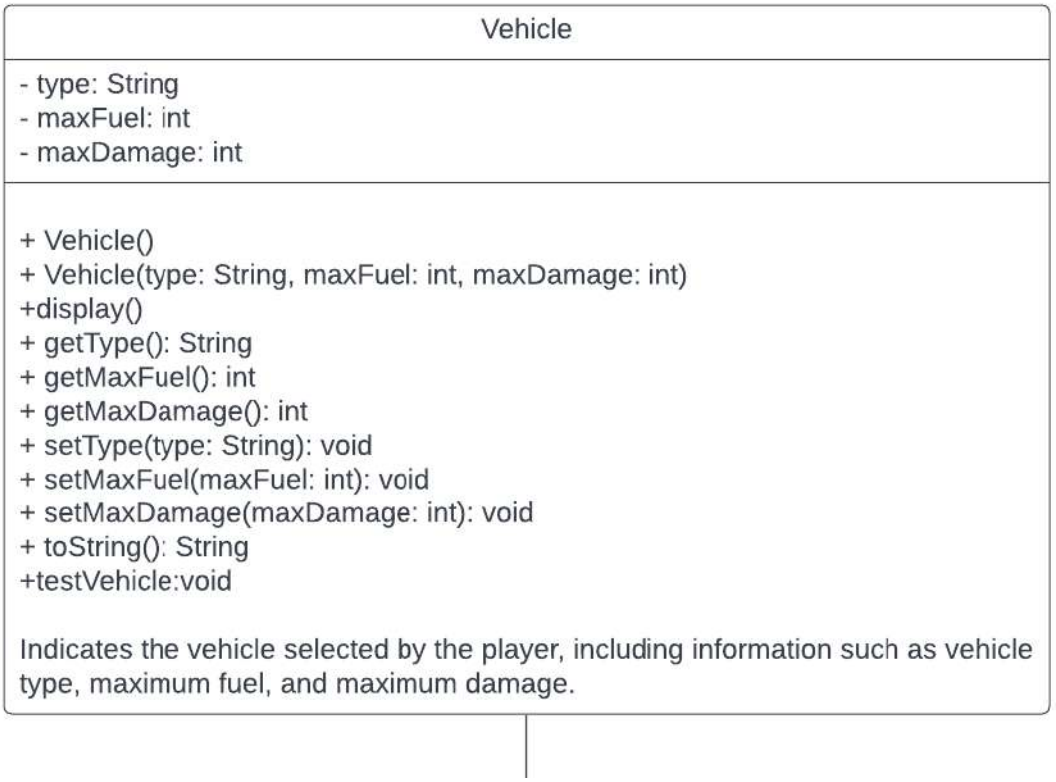
```
Validator initialized
Difficulty Level Information
Difficulty Option: Easy
Min Highway Length: 10
Max Highway Length: 15
Fuel Capacity: 1.0
Obstacle Count: 12
Current Highway Length: 13
```

```
Type:Default
Max Fuel:100
Max Damage:100
```

```
public void testPlayer()
    {
        System.out.println("Create an Player object with the
        Player player1 = new Player();
        player1.display();

        System.out.println("Create an Player object withe the
        Player player2 = new Player();
        player2.setPlayerName("ziqi");
        player2.setSelectedVehicle(new Vehicle());
        player2.setCurrentFuel(50);
        player2.setCurrentDamage(20);
        player2.setPlayerPositionX(2);
        player2.setPlayerPositionY(0);
        player2.display();

        System.out.println("Create an Player object withe the
        Player player3 = new Player();
        player3.setPlayerName("Joohn");
```

```
        player3.setSelectedVehicle(null);
        player3.setCurrentFuel(-50);
        player3.setCurrentDamage(-20);
        player3.setPlayerPositionX(-2);
        player3.setPlayerPositionY(0);
        player3.display();

    }
```

Test Vehicle Plan



1. Test the default constructor

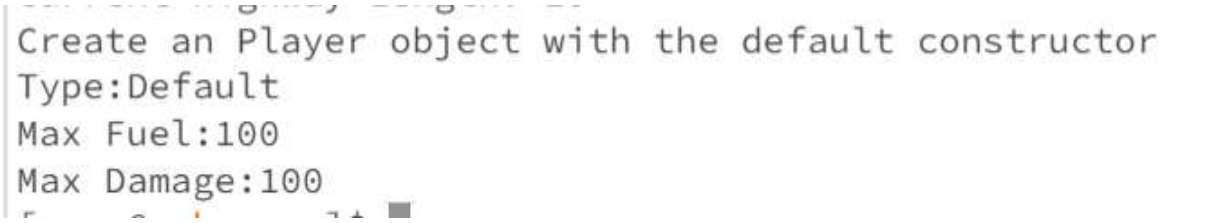
Except output

Type: defult

Max Fuel: 100

Max Damage:100

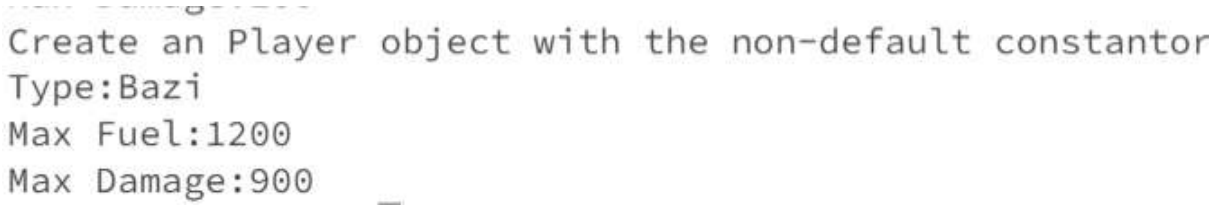
actual output



2. Test the non-default constructor

Excepted output

actual output

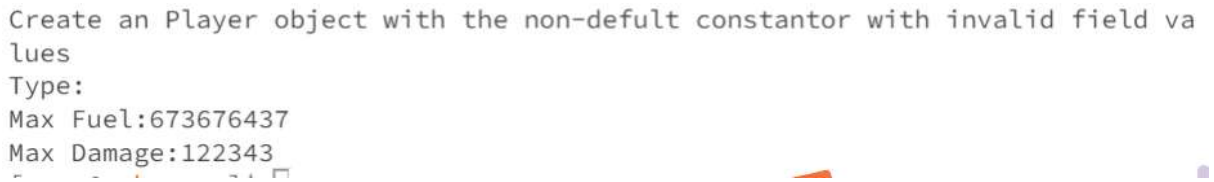


3. Test the non-default constructor with invalid values

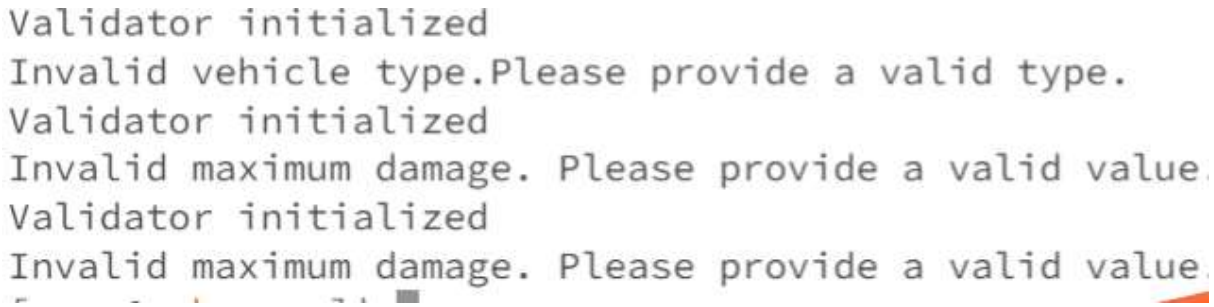
Expected output

Type:""
MaxFuel:676766437

MaxDamage:122343

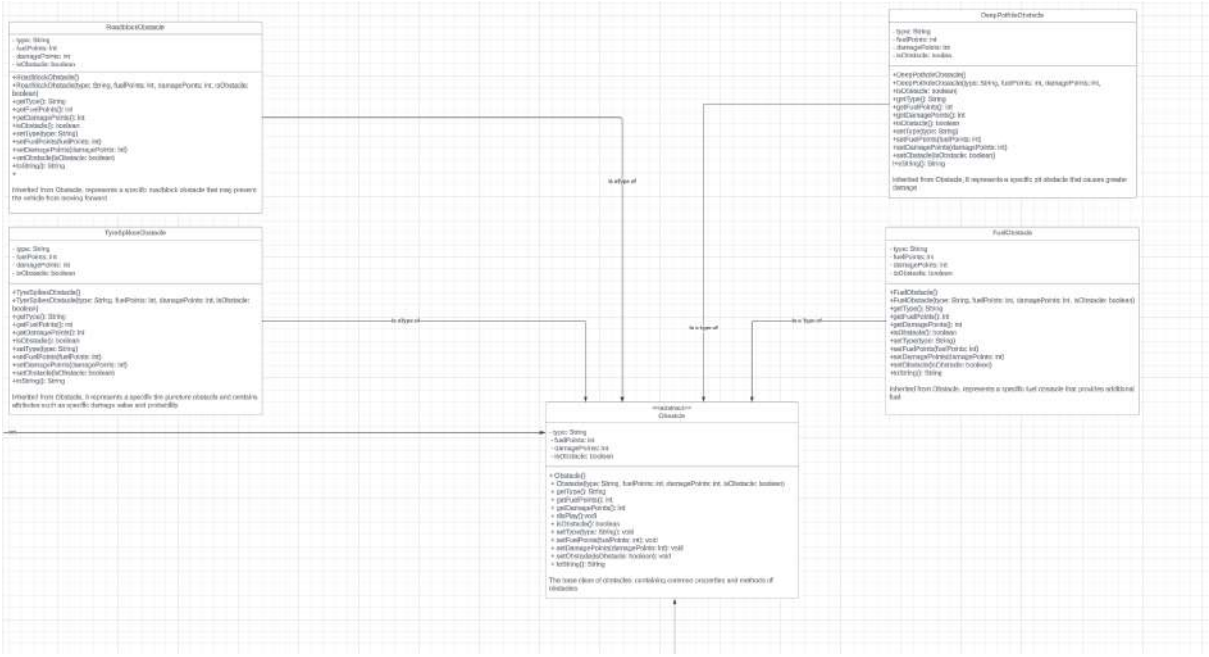


4. Test get-set values (Test succe



```
public void testVehicle()
    {
        System.out.println("Create an Vehicle object with the
        Vehicle vehicle1 = new Vehicle();
        vehicle1.display();

        System.out.println("Create an Vehicle object with the
        Vehicle vehicle2 = new Vehicle("Bazi",1200,900);
        vehicle2.display();

        System.out.println("Create an Vehicle object with the
        Vehicle vehicle3 = new Vehicle("", 673676437, 122343)
        vehicle3.display();

        System.out.println("test function use get seter");
        Vehicle vehicle4 = new Vehicle();
        vehicle4.setType("");
        vehicle4.setMaxFuel(673676437);
        vehicle4.setMaxDamage(122343);
    }
```

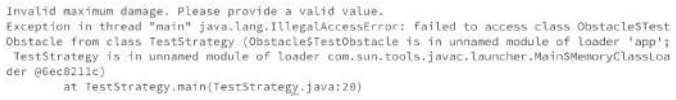
Test Obstacle



actual output

Test Deep Pothole Obstacle

- Expected Output:



Change code Obstacle cannot Initialize

```
Type: P
Fuel Points: 0
Damage Points: 60
Is Obstacle: true
DeepPotholeObstacle{type=P',fuelPoints=0, damagePoints=60, isObstacle=true}
Type: S
Fuel Points: 0
Damage Points: 45
Is Obstacle: true
TyreSpikesObstacle{type=S',fuelPoints=0, damagePoints=45, isObstacle=true}
Type: B
Fuel Points: 0
Damage Points: 0
Is Obstacle: true
RoadblockObstacle{type=B',fuelPoints=0, damagePoints=0, isObstacle=true}
Type: F
Fuel Points: 10
Damage Points: 0
Is Obstacle: true
FuelObstacle{type=F',fuelPoints=10, damagePoints=0, isObstacle=true}
```

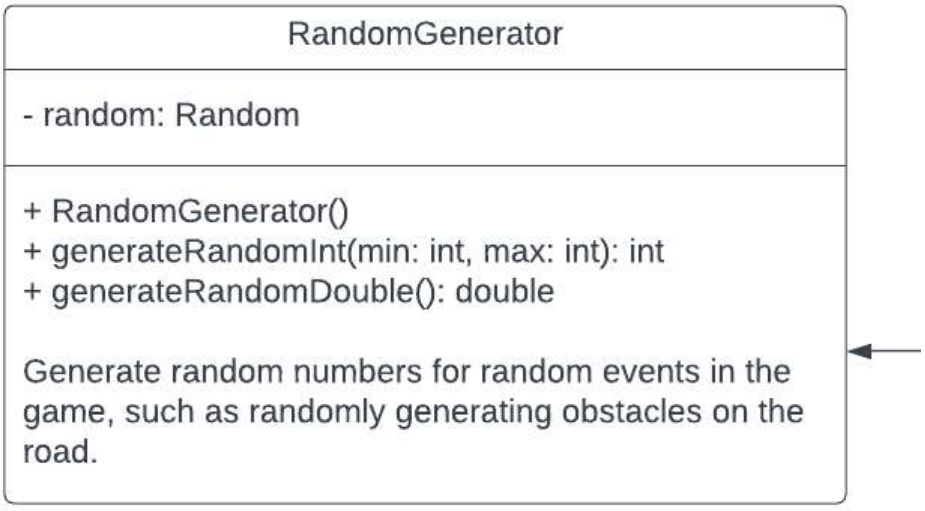
test code

```
public void testObstacle()
    {
        Obstacle obstacle1 = new DeepPotholeObstacle();
        obstacle1.display();
        System.out.println(obstacle1.toString());

        Obstacle obstacle2 = new TyreSpikesObstacle();
        obstacle2.display();
        System.out.println(obstacle2.toString());

        Obstacle obstacle3 = new RoadblockObstacle();
        obstacle3.display();
```

```
        System.out.println(obstacle3.toString());

        Obstacle obstacle4 = new FuelObstacle();
        obstacle4.display();
        System.out.println(obstacle4.toString());

    }
```

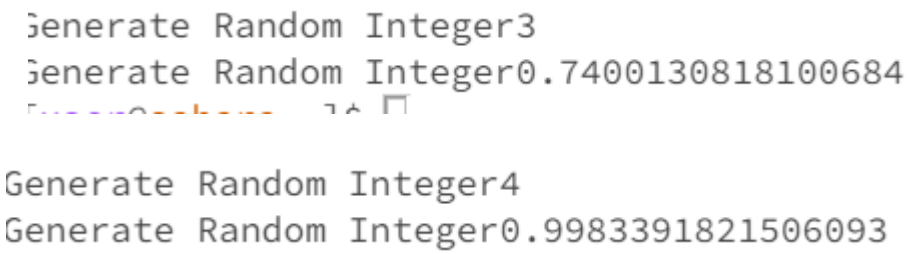
Test Generator

1. Test Minimum Length



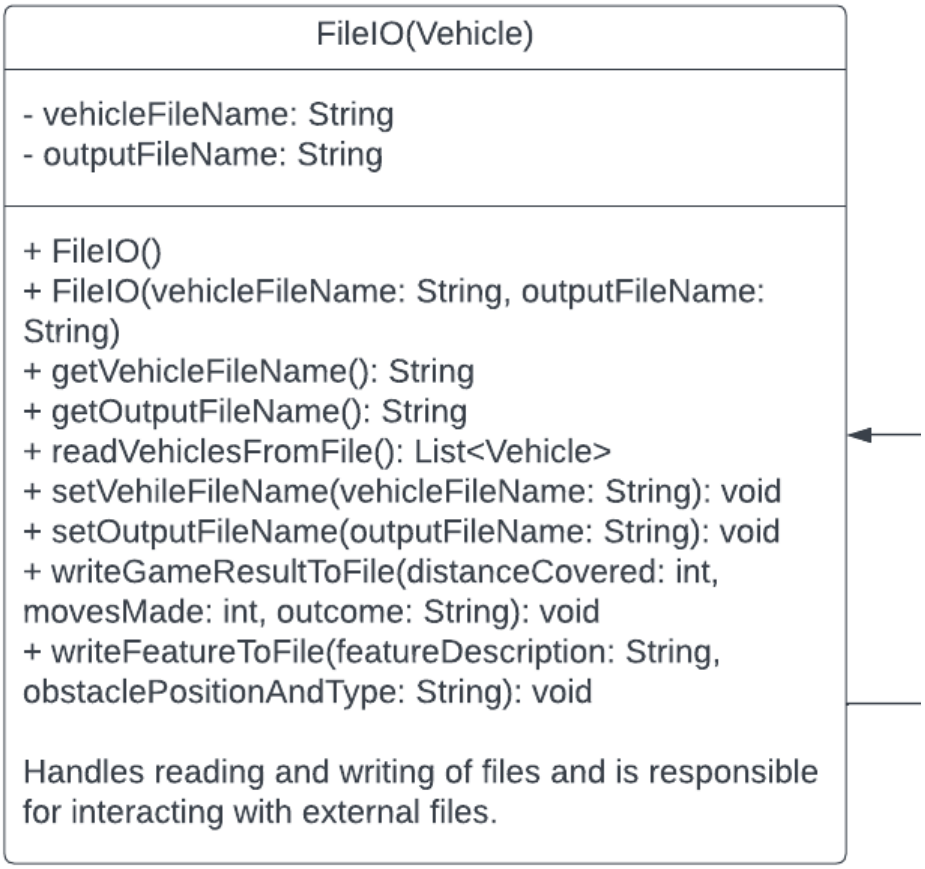
Test Random Output

want output minLenth max Length random
actual output



```
public void testRandomGenerator()
    {
        RandomGenerator randomGenerator = new RandomGenerator
        int randomInt = randomGenerator.generateRandomInt(2, 
        System.out.println("Generate Random Integer" + random

       RandomGenerator randomGenerator1 = new RandomGenerator
        double randomDouble = randomGenerator1.generateRandom
        System.out.println("Generate Random Integer" + random
    }
```

TestFielIO



Test FileIO

1. Test Default Constructor

- Actual Output:

```
Create an FielIO object with the default constantor
Vehicle File Name: vehicles.txt
Output File Name: output.txt
```

2. Test `readVehiclesFromFile`
    - Description: Read Vehicles from file (`output.txt`)

```
Create an FielIO object with the readVehiclesFromFile
Vehicle read from file:
Vehicle{Type='Motorcycle', maxFuel=100, maxDamage=30}
Vehicle{Type='Car', maxFuel=120, maxDamage=50}
Vehicle{Type='Bus', maxFuel=150, maxDamage=100}
```

```
Distance Covered: 100
Moves Made: 5
Outcome: Win
Distance Covered: 100
Moves Made: 5
Outcome: Win
Distance Covered: 100
Moves Made: 5
Outcome: Win
Distance Covered: 100
Moves Made: 5
Outcome: Win
Distance Covered: 100
Moves Made: 5
Outcome: Win
```

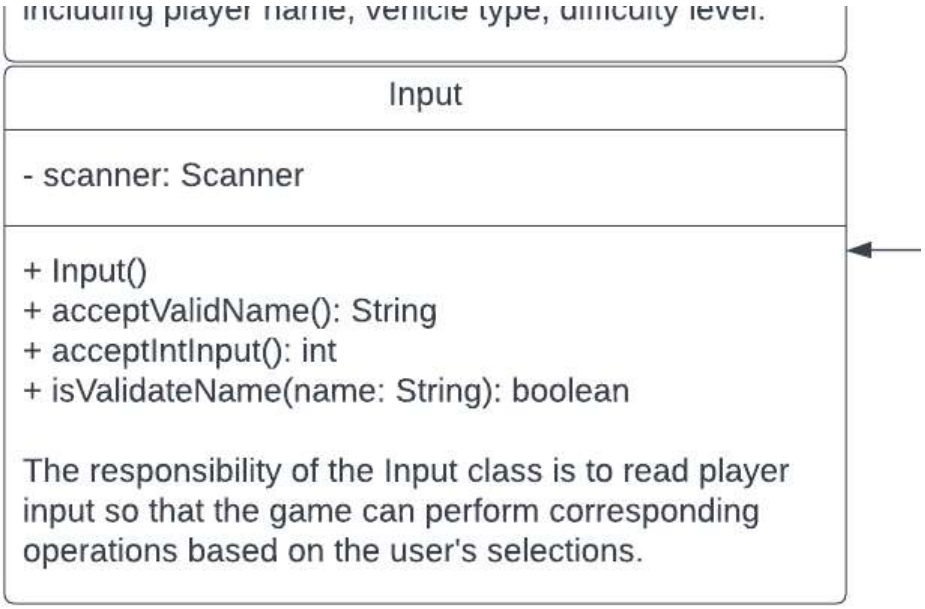
3. Test `writeFileToFile` (`feature.txt`)
    - Expected Output:

```
System.out.println("Test writing game result to file")
FileIO fileIO2 = new FileIO();
fileIO2.writeGameResultToFile(100, 5, "Win");
System.out.println("Game result written to file.");
```

```
1 Game Feature Description
2 obtstacle Position and Type
```

```
public void testFileIO()
    {
        System.out.println("Create an FielIO object with the
        FileIO fileIO = new FileIO();
        System.out.println("Vehicle File Name: " + getVehicle
        System.out.println("Output File Name: " + getOutputFi

        System.out.println("Create an FielIO object with the
        FileIO fileIO1 = new FileIO();
        List<Vehicle> vehicles = fileIO1.readVehiclesFromFile
        System.out.println("Vehicle read from file:");
        for(Vehicle vehicle : vehicles)
        {
            System.out.println(vehicle);
        }

        System.out.println("Test writing game result to file"
        FileIO fileIO2 = new FileIO();
        fileIO2.writeGameResultToFile(100, 5, "Win");
        System.out.println("Game result written to file.");

        System.out.println("Test write Feature To File ");
        fileIO2.writeFeatureToFile("Game Feature Description"
        System.out.println("Feature written to file.");


    }
```

Test Input



t

Test Input

1. Test Default Output
    - Expected Output:

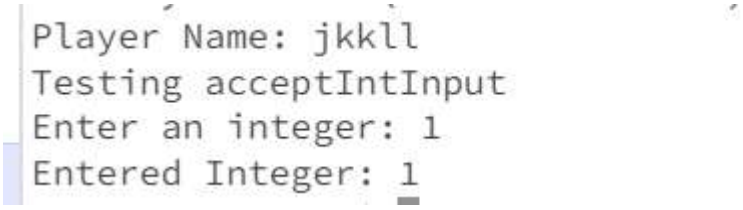
```
Testing acceptValidName:
Enter your name (5-10 characters, lowercase only): dsdsd
Player Name: dsdsd
```

2. Test `intInput`

- Description: Test the `intInput` method
- Expected Output:

```
Player Name :lowcase name with 5 character
interger : 1
```

- Actual Output:test pass

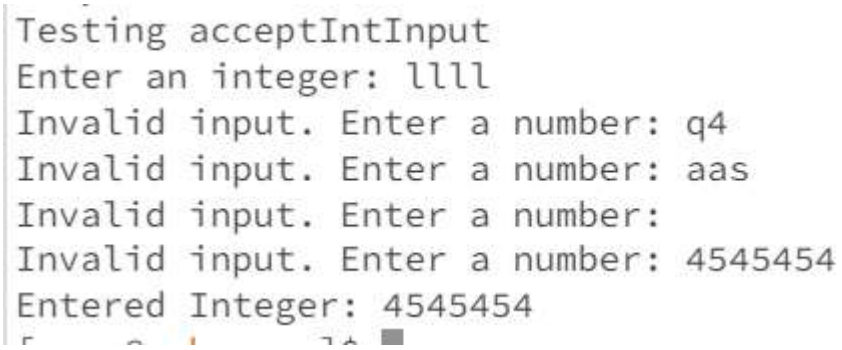


Invalid test

Expected Output:

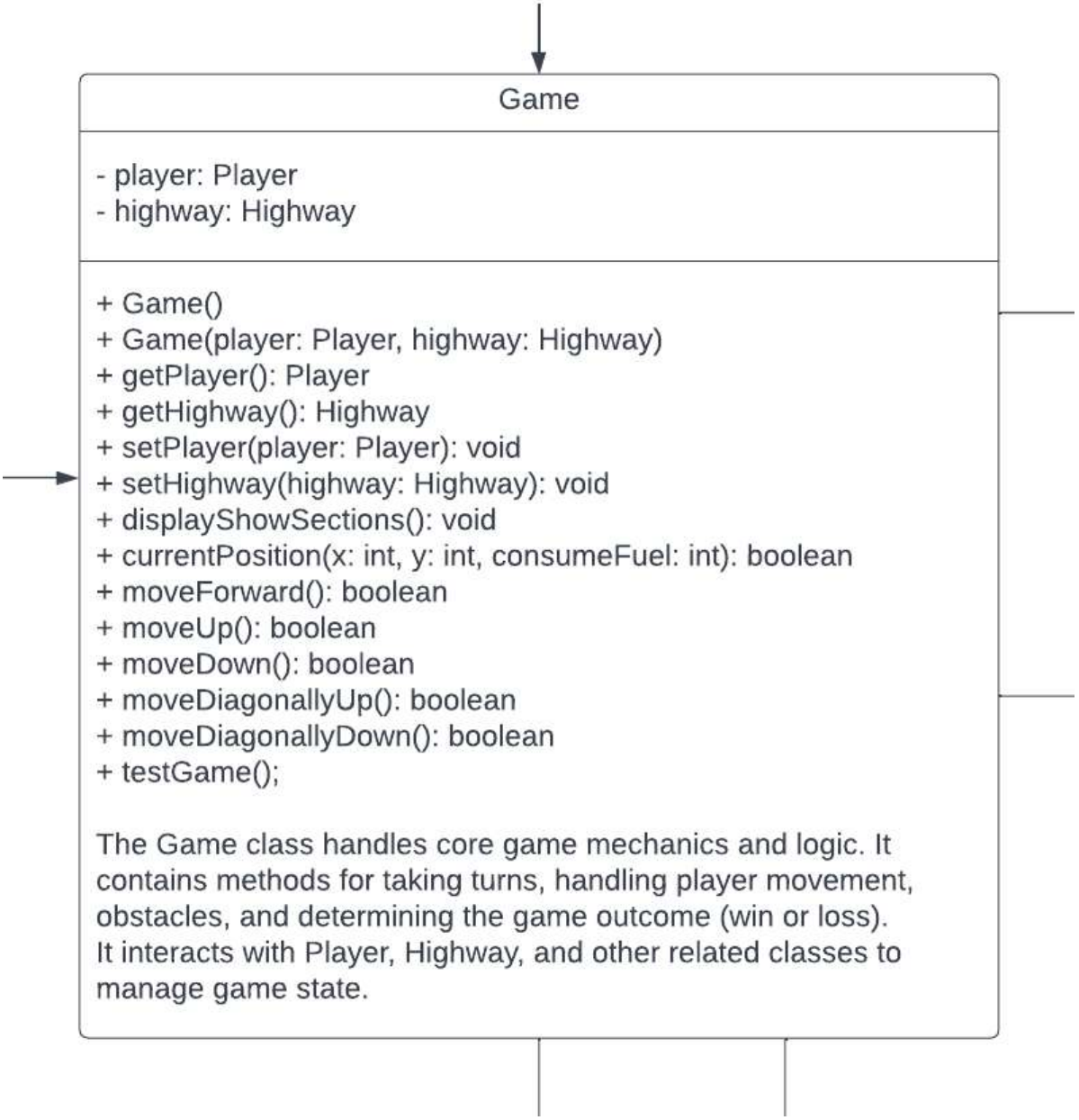
```
with name q and 4  Invalid input
with 3character Invalid input
with empty character Invalid input
```

- Actual Output:test pass



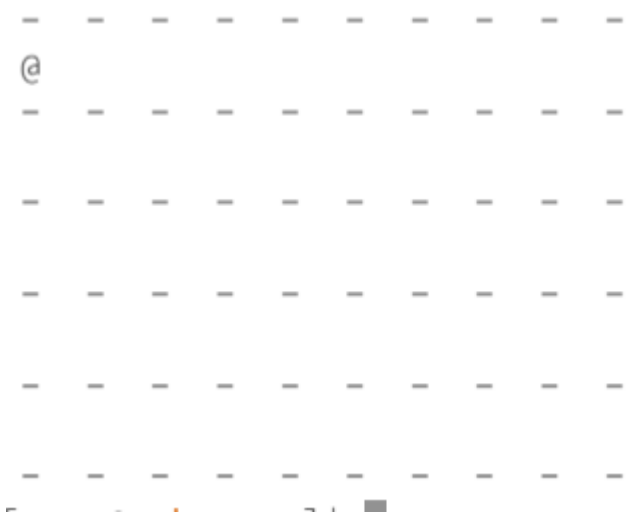
test pass

Game

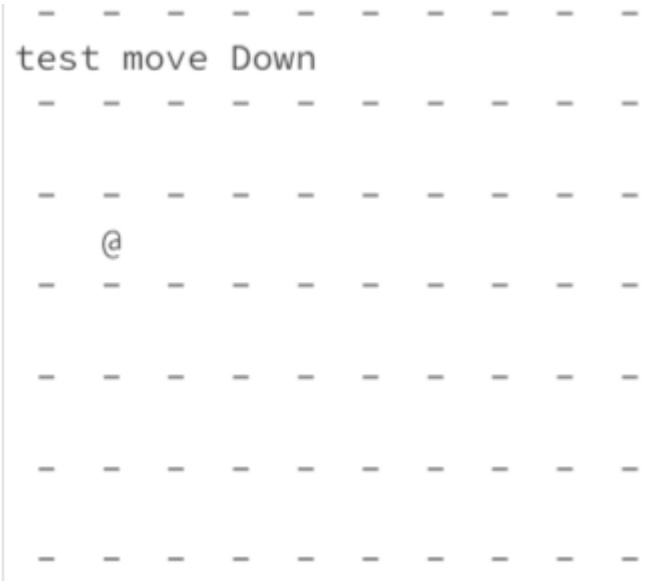


1. Test `displayShowSection`
    - Description: Show section with 10 views. The player can be represented using the character '@'

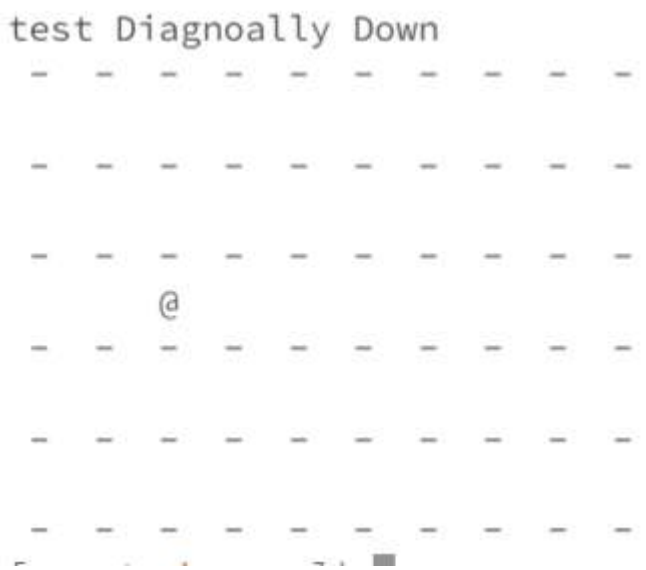
actual output



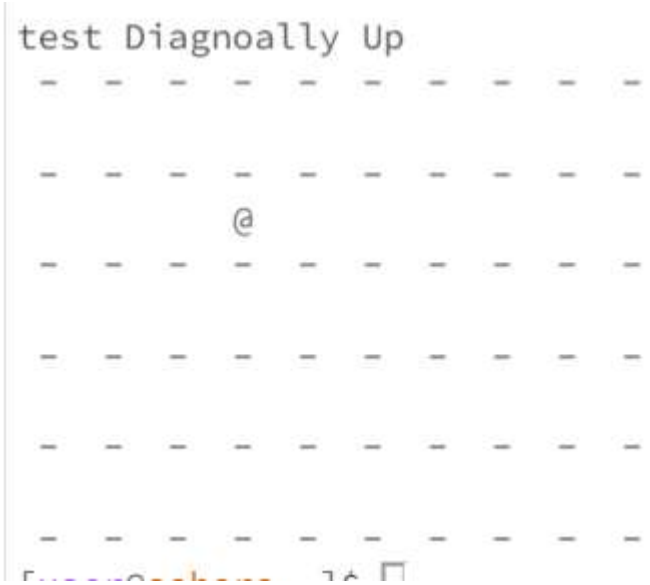
3. Test `moveDown`
    - Description: Character '@' move down
    - Expected Output:



4. Test `moveDiagonallyDown`
    - Description: Character '@' move diagonally down



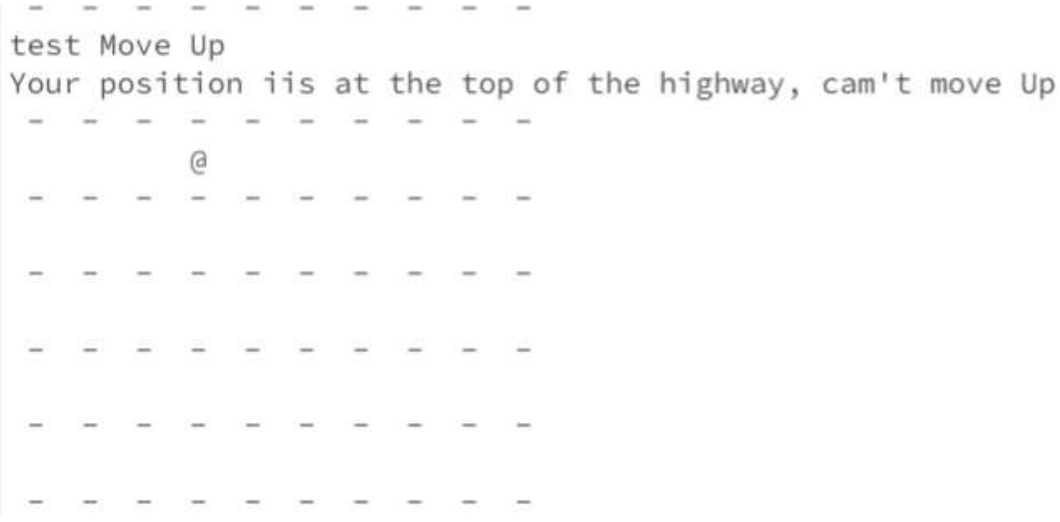
5. Test `moveDiagonallyUp`
    - Description: Character '@' move diagonally up



6. Test `moveUp`
    - Description: Character '@' move up

7. Boundary Test `moveUp` - Passed Again

- Description: Test moving up beyond the boundary



8. Test `updatePlayerStatusAtCurrentPosition`

- Description: Test updating player status at the current position

8. add

```
game3.getHighway().getSectionLanes()[1][3] = 'B';
```

test when meet B is can not move further

```
This obstacle can't be passed ; please try again.
Cannot move further: true
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
       @  B
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
Player's Current Fuel: 0
```

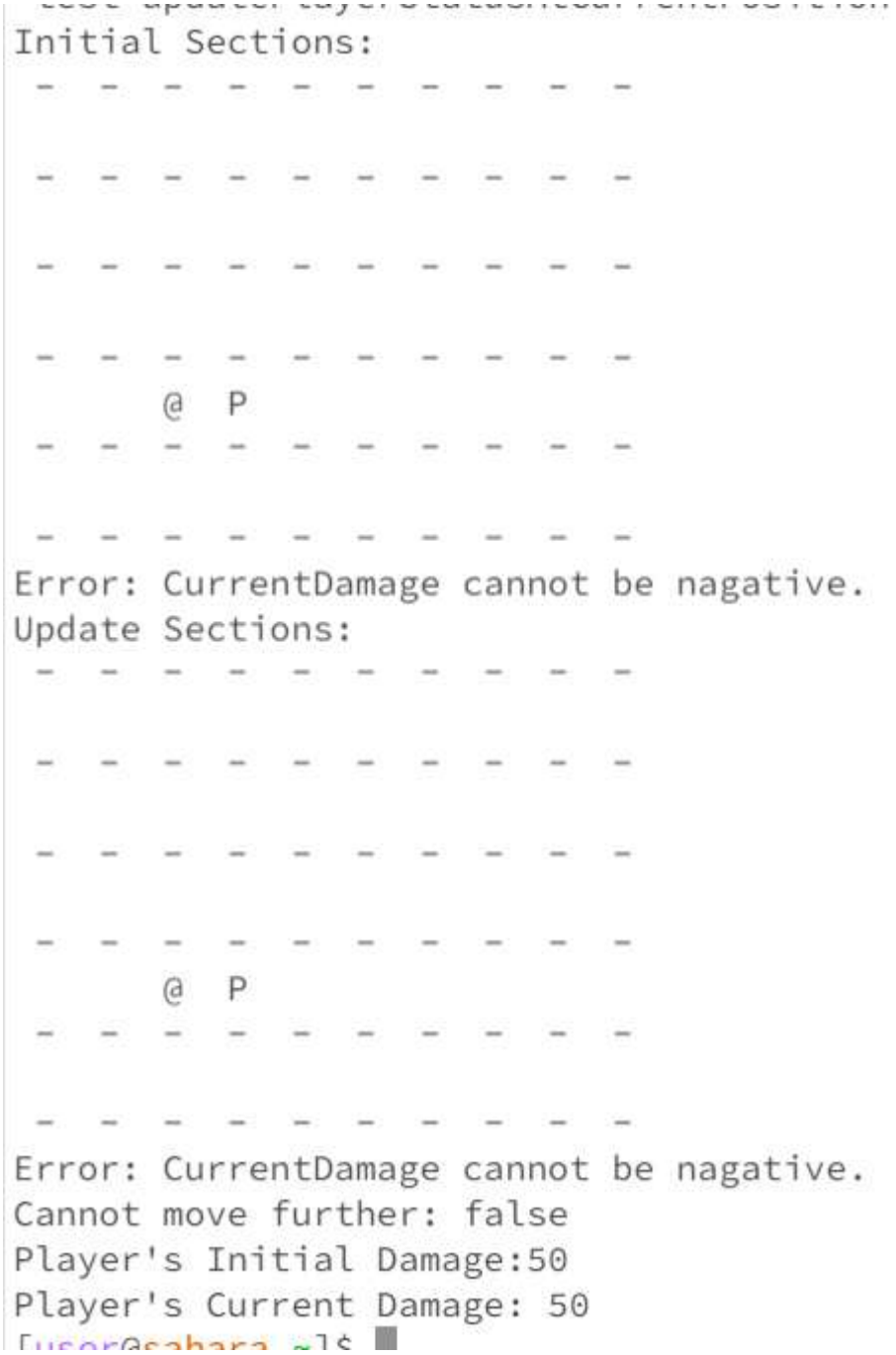
problem found at PlayersetCurrentFuel

```
public void setCurrentFuel(int currentFuel)
    {
        //If selectdVehicle is not null the value of this.cur
        //If selectedVehicle is null, the value of this curre
        this.currentFuel = (selectedVehicle != null ? selecte
    }
```

If the fuel is full when encountering obstacles, it will still be refueled.(false)

```
int fuel = (currentFuelAddTen > player.getSelectedVehicle().g
```

test 'P’



now output Current Damage is update

```
 test updatePlayerStatusAtCurrentPosition
Initial Sections:
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
       @  P
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
Update Sections:
Error: CurrentDamage cannot be nagative.
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
          @
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
Cannot move further: false
Player's Initial Damage:100
Player's Current Damage: 40
```

test code when meet S

```
 test updatePlayerStatusAtCurrentPosition S
Initial Sections:
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
       @  S
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
Update Sections:
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
          @
 -  -  -  -  -  -  -  -  -  -

 -  -  -  -  -  -  -  -  -  -
Cannot move further: false
Player's Initial Damage:100
Player's Current Damage: 55
```

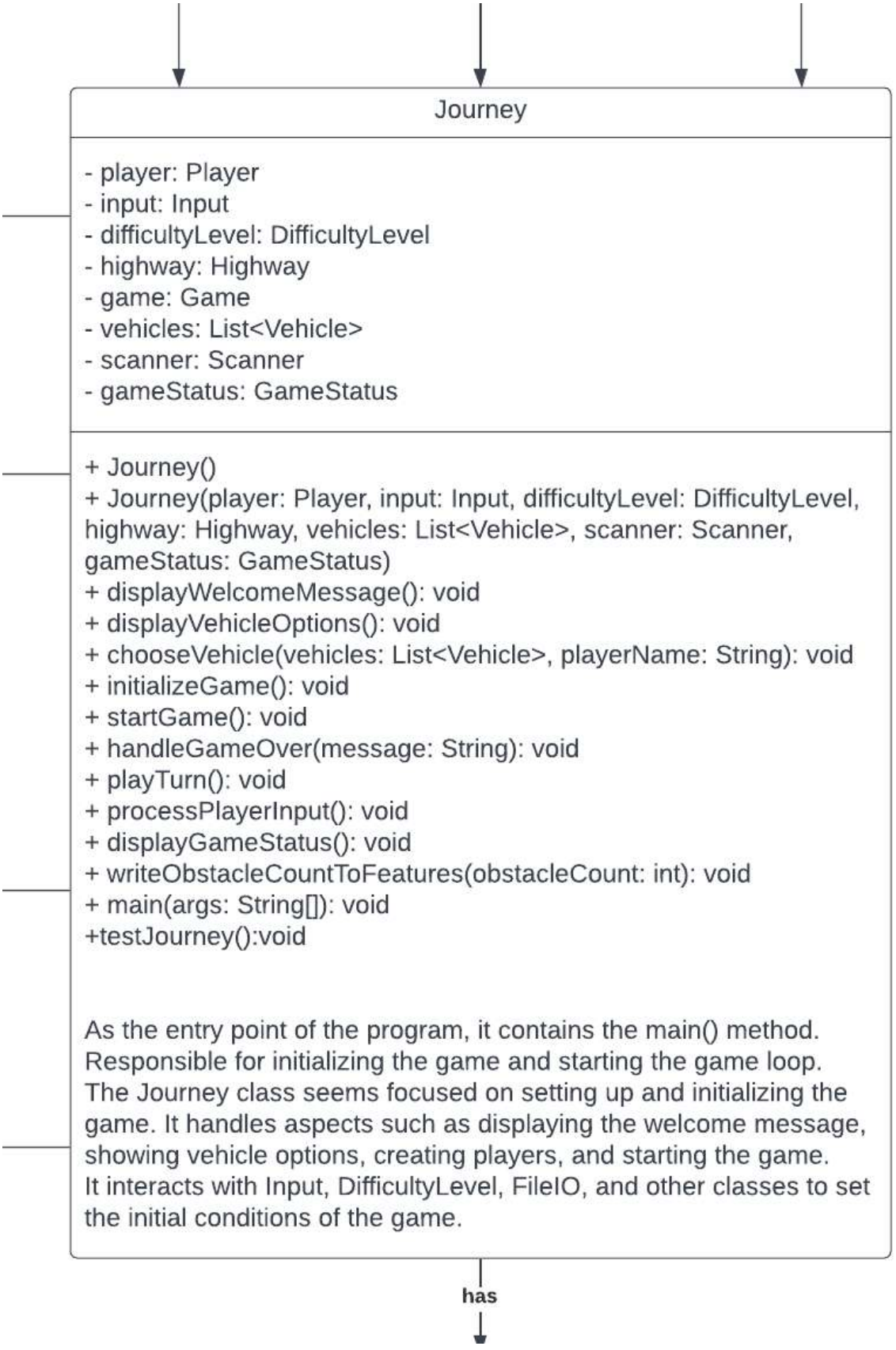
```
public void testGame()
    {
        // System.out.println("Create an object with show sec
        // Game game1 = new Game();
        // game1.displayShowSections();

        // System.out.println("Test Move Forward");
        // Game game2 = new Game();
        // game2.moveForward();
        // game2.displayShowSections();
```

```
        // System.out.println("test move Down");
        // game2.moveDown();
        // game2.displayShowSections();

        // System.out.println("test Diagnoally Down");
        // game2.moveDiagonallyDown();
        // game2.displayShowSections();

        // System.out.println("test Diagnoally Up");
        // game2.moveDiagonallyUp();
        // game2.displayShowSections();

        // System.out.println("test Move Up");
        // game2.moveUp();
        // game2.displayShowSections();

        // System.out.println("test Move Up");
        // game2.moveUp();
        // game2.displayShowSections();

        // System.out.println("Create an object with show sec
        // Game game3 = new Game();
        // game3.displayShowSections();
        // //Set the player's initial poistion
        // game3.getPlayer().setPlayerPositionX(1);
        // game3.getPlayer().setPlayerPositionY(2);
        // game3.player.setCurrentFuel(10);
        // //Test the updatePlayerStatusAtCurrentPosition met
        // System.out.println("Test Update Player Status At C
        // game3.getHighway().getSectionLanes()[1][3] = 'B';
        // System.out.println("Player's Current Fuel before u
        // boolean cannotMove = game3.updatePlayerStatusAtCur
        // //DisPlay the update sections and player's update
        // System.out.println("Player's Current Fuel: " + gam
        // System.out.println("Cannot move further: " + canno
        // System.out.println("Player's Current Damage: " + g
        // game3.displayShowSections();
```

```
        // System.out.println(" test updatePlayerStatusAtCurr
        // Game game4 = new Game();
        // Vehicle selectedVehicle = new Vehicle("Car", 100, 
        // game4.getPlayer().setSelectedVehicle(selectedVehic

        // //Set the player's initial poistion
        // game4.getPlayer().setCurrentFuel(10);
        // game4.getPlayer().setPlayerPositionX(1);
        // game4.getPlayer().setPlayerPositionY(1);
        // //Test the updatePlayerStatusAtCurrentPosition met
        // System.out.println("Initial Sections: ");
        // game4.displayShowSections();
        // FuelObstacle fuelObstacle = new FuelObstacle();
        // int fuelPoints = fuelObstacle.getFuelPoints();
        // boolean cannotMove2 = game4.updatePlayerStatusAtCu
        // //DisPlay the update sections and player's update 
        // System.out.println("Update Sections:");
        // game4.displayShowSections();
        // System.out.println("Player's Current Fuel: " + gam
        // System.out.println("Cannot move further: " + canno


        // System.out.println(" test updatePlayerStatusAtCurr
        // Game game6 = new Game();
        // Vehicle selectedVehicle3 = new Vehicle("Car", 100,
        // game6.getPlayer().setSelectedVehicle(selectedVehic
        // //Set the player's initial poistion
        // game6.getPlayer().setCurrentDamage(100);
        // game6.getPlayer().setPlayerPositionX(3);
        // game6.getPlayer().setPlayerPositionY(2);
        // //Test the updatePlayerStatusAtCurrentPosition met
        // game6.getHighway().getSectionLanes()[3][3] = 'P';
        // System.out.println("Initial Sections: ");
        // game6.displayShowSections();
        // DeepPotholeObstacle deepPotholeObstacle = new Deepl
        // int damagePointsP = deepPotholeObstacle.getDamageP
        // boolean cannotMoveP = game6.updatePlayerStatusAtCu
```

```
        // //DisPlay the update sections and player's update
        // System.out.println("Update Sections:");
        // game6.moveForward();
        // game6.displayShowSections();
        // System.out.println("Cannot move further: " + canno
        // System.out.println("Player's Initial Damage:100");
        // System.out.println("Player's Current Damage: " + g

        // System.out.println(" test updatePlayerStatusAtCurr
        // Game game7 = new Game();
        // Vehicle selectedVehicle4 = new Vehicle("Car", 100,
        // game7.getPlayer().setSelectedVehicle(selectedVehic
        // //Set the player's initial poistion
        // game7.getPlayer().setCurrentDamage(100);
        // game7.getPlayer().setPlayerPositionX(3);
        // game7.getPlayer().setPlayerPositionY(2);
        // //Test the updatePlayerStatusAtCurrentPosition met
        // game7.getHighway().getSectionLanes()[3][3] = 'S';
        // System.out.println("Initial Sections: ");
        // game7.displayShowSections();
        // boolean cannotMoveS = game7.moveForward();
        // //DisPlay the update sections and player's update
        // System.out.println("Update Sections:");
        // game7.displayShowSections();
        // System.out.println("Cannot move further: " + canno
        // System.out.println("Player's Initial Damage:100");
        // System.out.println("Player's Current Damage: " + g
    }
```

Test Journey



first want test displayWelcomeMessage();

```
Welcome to Java Journey
=======================
```

test readfile to vehicles.txt

```
Available Vehicles
```

test player.setPlayerName

```
Enter your name (5-10 characters, lowercase only): sdsdsd
```

```
Enter your name (5-10 characters, lowercase only): wqwq
Enter your name (5-10 characters, lowercase only): qwqw
Enter your name (5-10 characters, lowercase only): dsdsdsdssdd
Enter your name (5-10 characters, lowercase only): w w w w w w
Enter your name (5-10 characters, lowercase only): 12121212
Enter your name (5-10 characters, lowercase only): wewewe
Select difficulty
```

```
public boolean isValidateName(String name)
{
    return name.length() >= 5 && name.length() <= 10 && name.matches("^[a-z]+$");
}
```

```
Enter your name (5-10 characters, lowercase only): qwqwqwqqwqwqw
Invalid name. Please enter a name between 5 and 10 characters with only lowcase alphabetic characters.
Enter your name (5-10 characters, lowercase only): 11111111
Invalid name. Please enter a name between 5 and 10 characters with only lowcase alphabetic characters.
Enter your name (5-10 characters, lowercase only): 666666
Invalid name. Please enter a name between 5 and 10 characters with only lowcase alphabetic characters.
Enter your name (5-10 characters, lowercase only): wwww e
Invalid name. Please enter a name between 5 and 10 characters with only lowcase alphabetic characters.
```

test difficultyLevel.setDifficultyLevel

```
Select difficulty
1. Easy
2. Moderate
3. Hard
Enter Your choice
```

test InitialLanes

```
Sections Length: 10
Number of Lanes: 5
Highway Layout:
Lane 1: - - - - - - - - - - -
Lane 2: - - - - - - - - - - -
Lane 3: - - - - - - - - - - -
Lane 4: - - - - - - - - - - -
Lane 5: - - - - - - - - - - -
```

testObstacle
wrong because didn’t invoke difficultyLevel

```
Highway Information:
Sections Length: 14
Number of Lanes: 3
Highway Layout:
Lane 1: - - -B- - -P-F-P- - -B- -F-
Lane 2: - - - - - - - - - - -P- - -
Lane 3: - - - - - - -F-F- - -B-F-F-
```

test Choose vehicle
want outputdisplay Vehicle size

The error shows, and does not show, maximum fuel allowed (max fuel), and maximum sustainable damage (max damage).

```
Available Vehicles
1. Motorcycle
2. Car
3. Bus
```

add function

```
1. `List<Vehicle> vehicles = new FileIO().readVehiclesFromFil
`This line creates a new `FileIO` instance, calls its `readVe

2. `if(!vehicles.isEmpty())`
    Check that the vehicle list is not empty and make sure th
```

3. `System.out.println("Available Vehicles");`
   Prints a message indicating that available vehicles will

4. `for(int i = 0; i < vehicles.size(); i++)`
   Start a loop that iterates through each vehicle in the li

5. `Vehicle vehicle = vehicles.get(i);`
   Get the `i`th vehicle from the list and assign it to the

6. `System.out.println((i + 1) + ". " + vehicle.getType());`
   Prints the vehicle's index in the list and its type.

7. `System.out.println(" Maximum Fuel: " + vehicle.getMaxFuel
   Prints the maximum fuel allowed for the current vehicle.

8. `System.out.println(" Maximum Damage: " + vehicle.getMaxDa
   Prints the maximum damage the current vehicle can sustain

9. The loop continues with the next iteration, or exits after

10. `else`
    If the vehicle list is empty, print a message indicating

11. `System.out.println("No vehicle available. Exiting the ga
     Prints a message indicating that no vehicles are availa

12. `System.exit(0);`

```
public void displayVehicleOptions()
{
    List<Vehicle> vehicles = new FileIO().readV

    if(!vehicles.isEmpty())
    {
        System.out.println("Available Vehicles"
        for(int i = 0; i < vehicles.size(); i++
        {
            System.out.println((i + 1) + ". " +
        }
    }
    else
    {
        System.out.println("No vehicle availabl
        System.exit(0);
```

Fianal Actual output

```
Available Vehicles
1.Motorcycle
 Maximum Fuel: 100
 Max Damage: 30
2.Car
 Maximum Fuel: 120
 Max Damage: 50
3.Bus
 Maximum Fuel: 150
 Max Damage: 100
Choose a vehicle 1-3):
1
You choose one vehicleMotorcycle.
```

but code didn't insert HighwayLength

```
You choose one vehicleMotorcycle.
Highway Length is0
Your current fuel is0
Your corrent obstacleCount is0
```

modify code with Class attributes minus Journey's instance variables

```
You choose one vehicleCar.
Highway Length is10
Your current fuel is120
Your corrent obstacleCount is12
```

test game.displayShowSections
wantoutput view display Sectionwith 10

Actualoutput

```
Let's startGame
 -  -  -  -  -  -  -  -  -  -
 @           P  S  F        S
 -  -  -  -  -  -  -  -  -  -
                   F  B  B
 -  -  -  -  -  -  -  -  -  -
                F        F
 -  -  -  -  -  -  -  -  -  -
```

test processPlayerInput()

```
Validator initialized
Choose your move:
1. Move forward
2. Move up
3. Move down
4. Move diagonally up
5. Move diagonally down
1
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "Game.moveForward()" because "
this.game" is null
        at Journey.processPlayerInput(Journey.java:273)
        at Journey.testJourney(Journey.java:361)
        at TestStrategy.main(TestStrategy.java:45)
```

add Lane this.
I removed the local variable game of type Game and used this.game directly, making

sure to use the game object that had been initialized in the initializeGame method.

```
this.game = new Game(player, highway);
```

```
Validator initialized
Choose your move:
1. Move forward
2. Move up
3. Move down
4. Move diagonally up
5. Move diagonally down
1
```

With the addition of loop statements, these two functions can be used when the conditions are not met.

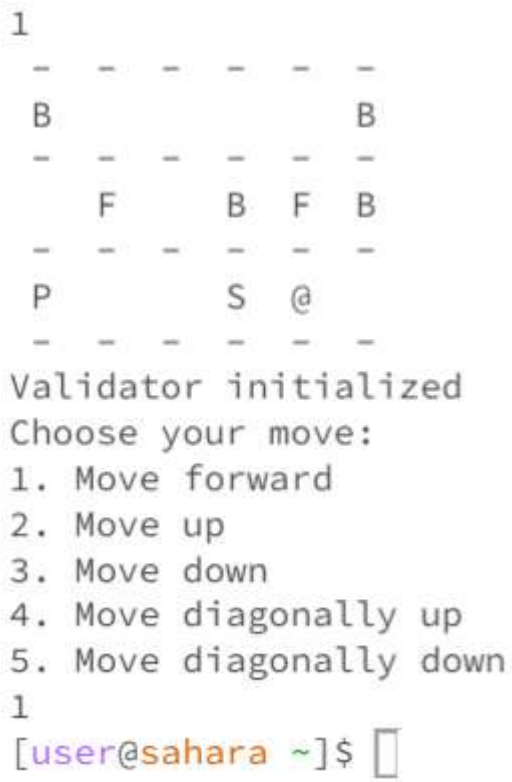
```
while(player.getPlayerPositionY() + 1 < highway.getSectionsLe
        {
            game.displayShowSections();
            processPlayerInput();
        }
```

add function when CurrentFuel < 0

```
if(player.getCurrentFuel() <= 0)
            {
                handleGameOver("The Vehicle didn't have fuel,
                break;
            }
```

add function whenChrrentDamage

want output Status when meet Damage




```
if(player.getCurrentDamage() <= 0)
            {
                handleGameOver("The Vehicle has been broken,
                break;
            }
```

After testing it many times I encountered many nails, but the CurrentDamage did not decrease

testboundary test

```
5. Move diagonally down
1
 -  -  -  -  -  -
 B              B
 -  -  -  -  -  -
    F     B  F  B
 -  -  -  -  -  -
 P        S  @
 -  -  -  -  -  -
Validator initialized
Choose your move:
1. Move forward
2. Move up
3. Move down
4. Move diagonally up
5. Move diagonally down
1
```

```
public void setCurrentDamage(int currentDamage)
{
    if(currentDamage >= 0)
    {
        this.currentDamage = currentDamage;
    }else
    {
        System.out.println("Error: CurrentDamage c
    }

}
```

Print the results in the penultimate section

Actual output

add print function when end loop

I found that many of them turned into B, which was not the gail I needed to generate.

```
Number of obstacle generated: 12
Generated obstacle at(1,4)of type:F
Generated obstacle at(0,9)of type:S
Generated obstacle at(2,9)of type:F
Generated obstacle at(2,5)of type:F
Generated obstacle at(1,6)of type:B
Generated obstacle at(1,7)of type:S
Generated obstacle at(0,8)of type:F
Generated obstacle at(1,10)of type:F
Generated obstacle at(1,5)of type:P
Generated obstacle at(0,7)of type:S
Generated obstacle at(1,9)of type:P
Generated obstacle at(2,3)of type:B
```

Verify initialization of obstacles:(true)
Double check the initializeObstacles method in the Highway class to make sure it generates obstacles correctly and places them on the lane. You can add print statements or use the debugger to inspect the generated obstacles.

```
Lane 1: - - - - -S- - - -B-S-B- - -
Lane 2: - - - - - - - -S-B- -S-S- -
Lane 3: - - - - -F- -B- - -F- -F- -
```

In the displayShowSections method of the Game class, add some additional print statements to output information about obstacles as each section is displayed.
Obstacle count same Gmae displayShow Section

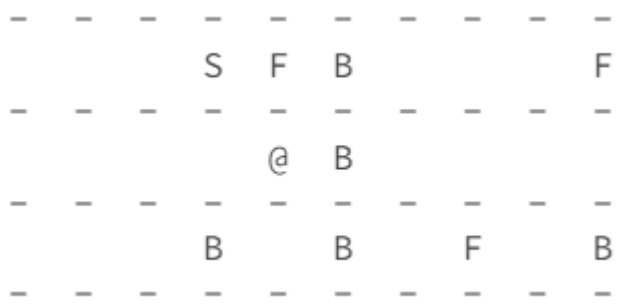
```
-  -  -  -  -  -  -  -  -
F        S
-  -  -  -  -  -  -  -  -
      P     @     S     F
-  -  -  -  -  -  -  -  -
   B  F     F  S  S
-  -  -  -  -  -  -  -  -
```

I need to add a special situation, that is, if you encounter B three times, it means you have entered a dead end, then you lose.


```
if(isInDeadEnd(player.getPlayerPositionX(), player.getPlayerP
            {
                handleGameOver("You encountered Roadblock thr
                break;
            }
```

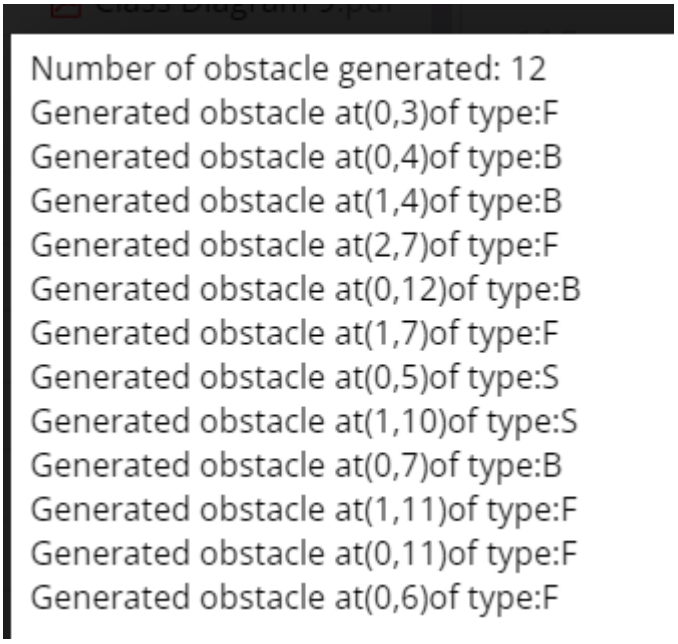
```
public boolean isInDeadEnd(int x, int y)
    {
        //check if the current position is'B';
        if(highway.getSectionLanes()[x][y] == 'B')
        {
            //check if the next section is also 'B'
            if(y + 1 < highway.getSectionsLength() && highway
            {
                //Check if the next sectio is also 'B'
                if(y + 2 < highway.getSectionsLength() && hig
                {
                    return true;// In a dead-end formed by th
                }
            }
        }
        return false;
    }
```

Encountered a situation where three B's are in a row, I need to re-disorganize



```
if(obstacleType == 'B')
                {
                    boolean allLanesOccupied = true;
                    //Rearrange the current lane if all posit
                    for(int i = 0; i < this.numLanes; i++)
                    {
                        if(sectionLanes[i][y] != 'B')
                        {
                            allLanesOccupied = false;
                            break;//No need to continue check
                        }
                    }
                    if(allLanesOccupied)
                    {
                        sectionLanes[x][y] = ' ';
                        continue;
                    }
                }
```

I discovered a very serious problem during testing, that is, the type and number of Obstacles generated at the beginning of my feature.txt were different from those actually generated.



obstaclePositionAndTypeList contains a list of position and type information of obstacles, and then appends the information of each obstacle to the obstaclePositionAndType string and writes it to the file through fileIO.writeFeatureToFile

```
List<String> obstaclePositionAndTypeList = highway.tellObstac
```

Create a function that is not called in InitialGame

```
public List<String> tellObstaclesInformation()
     {
        List<String> obstaclePositionAndTypeList = new ArrayL

        //iterate through each lane and section to identifity
        for(int x = 0; x < this.numLanes; x++)
        {
            for(int y = 0; y < this.sectionsLength; y++)
            {
                //Check if there is obstacle at the current p
                if(sectionLanes[x][y] != ' ')
```

```
                {
                    //Add obstacle information to the list
                    obstaclePositionAndTypeList.add("Generate
                }
            }
        }
        return obstaclePositionAndTypeList;
     }
```

write file to output.txt

```
1 Distance Covered: 8
2 Moves Made: 8
3 Outcome: The Vehicle has been broken, Game Over!
4 Distance Covered: 10
5 Moves Made: 10
6 Outcome: You got it, Win!!
```

write file to feature.txt

```
public void writeObstacleCountToFeatures(int obstacleCount)
    {
        //Create an instance of FileIO to handle file input a
        FileIO fileIO = new FileIO();
        //Create a String featureDescription that describes t
        String featureDescription = "Number of obstacle gener

        //Obtain the obstacle information list obstaclePositi
        List<String> obstaclePositionAndTypeList = highway.te

        // Use a StringBuilder object obstaclePositionAndType
        StringBuilder obstaclePositionAndType = new StringBui
        //By iterating over the obstaclePositionAndTypeList,a
        for(String obstacleInfomation : obstaclePositionAndTy
        {
```

```
            obstaclePositionAndType.append(obstacleInfomation
        }
        //Write feature description and obstacle position typ
        fileIO.writeFeatureToFile(featureDescription, obstacl
    }
```

```
Number of obstacle generated: 12
Generated obstacle at(0,3) of type: B
Generated obstacle at(0,4) of type: F
Generated obstacle at(0,7) of type: B
Generated obstacle at(0,8) of type: B
Generated obstacle at(0,9) of type: S
Generated obstacle at(1,5) of type: B
Generated obstacle at(1,7) of type: F
Generated obstacle at(2,3) of type: S
Generated obstacle at(2,6) of type: F
Generated obstacle at(2,7) of type: S
Generated obstacle at(2,8) of type: S
Generated obstacle at(2,9) of type: S

```

test code

```
public void testJourney()
    {
        // Journey journey = new Journey();
        // displayWelcomeMessage();

        // displayVehicleOptions();
        // player.setPlayerName(input.acceptValidName());
        // String playerName = player.getPlayerName();
        // difficultyLevel.setDifficultyLevel();

        // // choose the current HighwayLength - numLanes is
        // highway = new Highway(difficultyLevel.getCurrentHi
        // highway.initializeLanes();
```

```
        // highway.display();
        // int obstacleCount = difficultyLevel.getObstacleCou
        // this.highway.initializeObstacles(obstacleCount);//
        // highway.display();

        // chooseVehicle(vehicles, playerName);
        // this.game = new Game(player, highway);

        // System.out.println("Let the Game begin!");

        // // The main game loop that continues unitil the pl
        // while(player.getPlayerPositionY() + 1 < highway.ge
        // {
        //     //Check if the player runs out of fuel
        //     if(player.getCurrentFuel() <= 0)
        //     {
        //         handleGameOver("The Vehicle didn't have fu
        //         break;
        //     }
        //     //Check if the player's vehicle is broken
        //     if(player.getCurrentDamage() <= 0)
        //     {
        //         handleGameOver("The Vehicle has been broke
        //         break;
        //     }
        //     // Check if the player runs into Roadblock(B)
        //     if(game.isInDeadEnd(player.getPlayerPositionX(
        //     {
        //         handleGameOver("You encountered Roadblock
        //         break;
        //     }

        //     //Executed a turn in the game
        //     playTurn();

        //     //Increment the move count in the game status
        //     int moveTimes = gameStatus.getMoveTimes();
        //     gameStatus.setMoveTimes(moveTimes + 1);
```

```
        // }
        // game.displayShowSections();
        // //Check if the player reached the end of the highw
        // if(player.getPlayerPositionY() == highway.getSecti
        // {
        //     gameStatus.setGameResult("You got it, Win!!");
        // }

        // //Set the total drive distance in the game status
        // gameStatus.setDriveDistance(player.getPlayerPositi

        // //Display the final game status
        // displayGameStatus();

        // //Write the obstacle count to the feature file
        // writeObstacleCountToFeatures(difficultyLevel.getOb

        // //Close the scanner to avoid resoure leaks
        // scanner.close();

    }
```