

Tâche 1 : Créez la classe `Ship` dans le répertoire `src/main/java` de votre projet qui répond aux spécifications ci-dessus.

```
public class Ship {
    private int lifePoints;
    private final String name;

    public Ship(int length, String name) {
        this.lifePoints = length;
        this.name = name;
    }

    public boolean hasBeenSunk(){
        return getLifePoints() <= 0;
    }

    public int getLifePoints() {
        return lifePoints;
    }

    public void takeAHit(){
        if(lifePoints>0)
            lifePoints--;
    }

    @Override
    public String toString() {
        return name + " (" + lifePoints + " LP)";
    }
}
```

Tâche 2 : Créez une classe de test nommée `ShipTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests la spécification du comportement de la classe `Ship`.

```
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.*;

public class ShipTest {
    @Test
    public void testTakeAHit(){
        Ship submarine = new Ship(3, "Submarine");
        assertThat(submarine.getLifePoints()).isEqualTo(3);
    }
}
```

```

    submarine.takeAHit();
    assertThat(submarine.getLifePoints()).isEqualTo(2);
    Ship carrier = new Ship(5, "Carrier");
    assertThat(carrier.getLifePoints()).isEqualTo(5);
    carrier.takeAHit();
    carrier.takeAHit();
    assertThat(carrier.getLifePoints()).isEqualTo(3);
}

@Test
public void testTakeAHitWithZeroLifePoint(){
    Ship zeroLifeShip = new Ship(0, "Submarine");
    assertThat(zeroLifeShip.getLifePoints()).isEqualTo(0);
    zeroLifeShip.takeAHit();
    assertThat(zeroLifeShip.getLifePoints()).isEqualTo(0);
}

@Test
public void testGetLifePoints(){
    Ship submarine = new Ship(3, "Submarine");
    assertThat(submarine.getLifePoints()).isEqualTo(3);
    Ship carrier = new Ship(5, "Carrier");
    assertThat(carrier.getLifePoints()).isEqualTo(5);
}

@Test
public void testHasBeenSunk(){
    Ship oneLife = new Ship(1, "One life");
    assertThat(oneLife.hasBeenSunk()).isFalse();
    oneLife.takeAHit();
    assertThat(oneLife.hasBeenSunk()).isTrue();
    Ship submarine = new Ship(3, "Submarine");
    for(int index = 0; index < 3; index++) {
        assertThat(submarine.hasBeenSunk()).isFalse();
        submarine.takeAHit();
    }
    assertThat(submarine.hasBeenSunk()).isTrue();
}

@Test
public void testToString(){
    Ship oneLife = new Ship(1, "One life ship");
    assertThat(oneLife.toString()).isEqualTo("One life ship (1 LP)");
    Ship submarine = new Ship(3, "Submarine");
    assertThat(submarine.toString()).isEqualTo("Submarine (3 LP)");
}
}

```

Tâche 3 : Créez le type `ShotResult` dans le répertoire `src/main/java` de votre projet.

```
public enum ShotResult {  
    MISSED, HIT, SUNK  
}
```

Tâche 4 : Créez la classe `Cell` dans le répertoire `src/main/java` de votre projet qui répond aux spécifications ci-dessus.

```
public class Cell {  
    private Ship ship = null;  
    private boolean hasBeenShot = false;  
  
    public Ship getShip() {  
        return ship;  
    }  
  
    public void setShip(Ship ship) {  
        this.ship = ship;  
    }  
  
    public boolean hasBeenShot() {  
        return hasBeenShot;  
    }  
  
    public ShotResult takeAShot(){  
        if(hasBeenShot())  
            return ShotResult.MISSED;  
        hasBeenShot = true;  
        if(!hasShip())  
            return ShotResult.MISSED;  
        ship.takeAHit();  
        if(ship.hasBeenSunk())  
            return ShotResult.SUNK;  
        return ShotResult.HIT;  
    }  
  
    public boolean hasShip() {  
        return getShip() != null;  
    }  
}
```

Tâche 5 : Créez une classe de test nommée `CellTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests la spécification du comportement de la classe `Cell`.

Tâche 6 : Créez la classe `Coordinates` dans le répertoire `src/main/java` de votre projet qui répond aux spécifications ci-dessus.

```

import java.util.Objects;

public class Coordinates {
    private final int x;
    private final int y;

    public Coordinates(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(Object o) {
        if (!(o instanceof Coordinates)) return false;
        Coordinates coordinates = (Coordinates) o;
        return x == coordinates.x && y == coordinates.y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }

    public Coordinates add(Coordinates coordinates){
        return new Coordinates(this.x + coordinates.x, this.y + coordinates.y);
    }
}

```

Tâche 7 : Créez une classe de test nommée `CoordinatesTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests la spécification du comportement de la classe `Coordinates`.

Tâche 8 : Créez le type `Orientation` dans le répertoire `src/main/java` de votre projet.

```

public enum Orientation {
    VERTICAL, HORIZONTAL
}

```

Tâche 9 : Créez la classe `Position` dans le répertoire `src/main/java` de votre projet qui répond aux spécifications ci-dessus.

```
public class Position {
    private final int length;
    private final Orientation orientation;
    private final Coordinates firstCell;

    public Position(Ship ship, Coordinates firstCell, Orientation orientation) {
        this.length = ship.getLifePoints();
        this.orientation = orientation;
        this.firstCell = firstCell;
    }

    public Coordinates[] shipCoordinates(){
        Coordinates increment = (orientation == Orientation.HORIZONTAL) ?
            new Coordinates(1,0) : new Coordinates(0, 1);
        Coordinates coordinates = firstCell;
        Coordinates[] shipCoordinates = new Coordinates[length];
        for(int index = 0; index < length; index++){
            shipCoordinates[index] = coordinates;
            coordinates = coordinates.add(increment);
        }
        return shipCoordinates;
    }
}
```

Tâche 10 : Créez une classe de test nommée `PositionTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests la spécification du comportement de la classe `Position`.

Tâche 11 : Créez la classe `Grid` dans le répertoire `src/main/java` de votre projet. C'est à vous de définir les signatures des méthodes de cette classe ainsi que leur comportement.

```
public class Grid {
    private final int width;
    private final int height;
    private final Cell[][] cells;

    public Grid(int width, int height){
        this.width = width;
        this.height = height;
        cells = new Cell[width][height];
        for(int x = 0; x < width; x++){
            for(int y = 0; y < height; y++){
                cells[x][y] = new Cell();
            }
        }
    }
}
```

```

}

public int getWidth() {
    return width;
}

public int getHeight() {
    return height;
}

private Cell getCell(Coordinates coordinates){
    return cells[coordinates.getX()][coordinates.getY()];
}

public boolean contains(Coordinates coordinates){
    return 0 <= coordinates.getX()
        && coordinates.getX() < width
        && 0 <= coordinates.getY()
        && coordinates.getY() < height;
}

/**
 * Shoot at the {@code Cell} of the {@code Grid} at the specified {@code coordinates}
 * and return the result of the shot.
 *
 * If the specified coordinates are outside the range of the grid, the call of the method
 * only returns {@code ShotResult.MISSED}.
 *
 * For the first shot at some specific coordinates included in the {@code Grid}, the result
 * depends on the presence or not of a ship at the corresponding {@code Cell}. If there are
 * no ship, the result of the shot is always {@code ShotResult.MISSED}. If there is a ship
 * the result of the shot is {@code ShotResult.SUNK} if it was the last life point of the
 * ship and {@code ShotResult.SUNK} otherwise. For any subsequent shot, the result of the
 * shot is always {@code ShotResult.MISSED}.
 *
 * @param coordinates the coordinates of the target {@code Cell}
 * @return {@code ShotResult.MISSED}, {@code ShotResult.HIT} or {@code ShotResult.SUNK}
 * depending on the result of the shot.
 */
public ShotResult shootAt(Coordinates coordinates){
    if(!contains(coordinates))
        return ShotResult.MISSED;
    return getCell(coordinates).takeAShot();
}

public boolean hasBeenShotAt(Coordinates coordinates){
    if(!contains(coordinates))

```

```

        return false;
    return getCell(coordinates).hasBeenShot();
}
public boolean hasShipAt(Coordinates coordinates){
    if(!contains(coordinates))
        return false;
    return getCell(coordinates).hasShip();
}

public void putShip(Ship ship, Coordinates firstCell, Orientation orientation){
    Position position = new Position(ship, firstCell, orientation);
    for (Coordinates coordinates:position.shipCoordinates()){
        getCell(coordinates).setShip(ship);
    }
}
}
}

```

Tâche 12 : Créez une classe de test nommée `GridTest` dans le répertoire `src/test/java` de votre projet qui devra vérifier via des tests la spécification du comportement de la classe `Grid`.