```java
public class Direction {

  /* Constants */

  public static final int[] X_STEP = { 0,  1, 0, -1 };
  public static final int[] Y_STEP = { -1, 0, 1,  0 };
  public static final String[] NAME = { "Up", "Right", "Down", "Left" };

  public static final int UP =    0;
  public static final int RIGHT = 1;
  public static final int DOWN =  2;
  public static final int LEFT =  3;

  public static final int LEFT_TURN = -1;
  public static final int NO_TURN =    0;
  public static final int RIGHT_TURN = 1;

  /* Attributes */

  private int direction;

  public Direction(int direction) {
    assert(direction >= 0 && direction < 4);

    this.direction = direction;
  }

  /**
   * Turns by a relative amount.
   * Positive relative directions are clockwise turns.
   * Negative relative directions are counter-clockwise turns.
   * Zero relative directions give back the direction unchanged.
   **/
  public Direction turn(int relativeDirection) {
    int newDirection = (this.direction + relativeDirection) % 4;

    if(newDirection < 0)
      newDirection += 4;

    return new Direction(newDirection);
  }

  /**
   * Retrieves the change in x-position when moving one step in current directio
n.
   **/
  public int moveX() {
    return X_STEP[this.direction];
  }

  /**
   * Retrieves the change in y-position when moving one step in current directio
n.
   **/
  public int moveY() {
    return Y_STEP[this.direction];
  }

  /**
   * Produces a string representation for this direction.
   **/
  public String toString() {
    return NAME[this.direction];
  }

  public static Direction createUp() {
    return new Direction(UP);
  }

  public static Direction createRight() {
    return new Direction(RIGHT);
  }

  public static Direction createDown() {
    return new Direction(DOWN);
  }

  public static Direction createLeft() {
    return new Direction(LEFT);
  }

  public static void main(String[] args) {
    Direction dir_up    = new Direction(UP);     // Alt: Direction.createUp();
    Direction dir_right = new Direction(RIGHT); // Alt: Direction.createRight();
    Direction dir_down  = new Direction(DOWN);  // Alt: Direction.createDown();
    Direction dir_left  = new Direction(LEFT);  // Alt: Direction.createLeft();

    System.out.println("Orientations:");

    System.out.println(dir_up);
    System.out.println(dir_right);
    System.out.println(dir_down);
    System.out.println(dir_left);

    System.out.println();
    System.out.println("Movements:");

    System.out.print("Up    : X : ");
    System.out.println(dir_up.moveX());
    System.out.print("Up    : Y : ");
    System.out.println(dir_up.moveY());

    System.out.print("Right : X : ");
    System.out.println(dir_right.moveX());
    System.out.print("Right : Y : ");
    System.out.println(dir_right.moveY());

    System.out.print("Down  : X : ");
    System.out.println(dir_down.moveX());
    System.out.print("Down  : Y : ");
    System.out.println(dir_down.moveY());

    System.out.print("Left  : X : ");
    System.out.println(dir_left.moveX());
    System.out.print("Left  : Y : ");
    System.out.println(dir_left.moveY());
  }
}

/**
 Output:

 Orientations:
 Up
 Right
 Down
 Left

 Movements:
 Up    : X : 0
 Up    : Y : -1
 Right : X : 1
 Right : Y : 0
 Down  : X : 0
 Down  : Y : 1
 Left  : X : -1
 Left  : Y : 0
 */

import java.util.ArrayList;
import java.util.Scanner;

public class Snake {
  //
  // Instansvariabel snakeList - B1
  //

  private int gameWidth;
  private int gameHeight;

  private int fruitX;
  private int fruitY;

  private boolean gameOver;

  //
  // Konstruktor - B1
  //

  public boolean isDone() {
    return snakeList.size() == gameWidth * gameHeight;
  }

  public int score() {
    return snakeList.size() - 1;
  }

  //
  // Metod placeFruit - B2
  //

  public boolean collideWithSnake(int x, int y) {
    for(SnakePart p : snakeList) {
      if(p.testPosition(x, y)) {
        return true;
      }
    }

    return false;
  }

  public boolean move(int relativeDirection) {
    SnakePart newHead = snakeList.get(0).move(relativeDirection);

    if(!newHead.isInside(this.gameWidth, this.gameHeight)) {
      gameOver = true;
      return false;
    }

    int newHeadX = newHead.getX();
    int newHeadY = newHead.getY();

    //
    // B3
    //

    return true;
  }

  public boolean isGameOver() {
    return this.gameOver;
```

```
  }

  @Override
  public String toString() {
    char [][] gameDisplay = new char[gameWidth][gameHeight];

    for(int x = 0; x < gameWidth; ++x) {
      for(int y = 0; y < gameHeight; ++y) {
        gameDisplay[x][y] = '-';
      }
    }

    gameDisplay[fruitX][fruitY] = 'X';

    // Draw the head of the snake
    gameDisplay[snakeList.get(0).getX()][snakeList.get(0).getY()] = '@';

    // Draw the tail of the snake
    for(int i = 1; i < snakeList.size(); ++i) {
      SnakePart p = snakeList.get(i);

      int px = p.getX();
      int py = p.getY();

      gameDisplay[px][py] = 'O';
    }

    String s = "";

    for(int y = 0; y < gameHeight; ++y) {
      for(int x = 0; x < gameWidth; ++x) {
        s = s + gameDisplay[x][y];
      }
      s = s + '\n';
    }

    return s;
  }

  public static void main(String[] args) {
    Snake s = new Snake(12, 8, 5, 4, new Direction(Direction.LEFT));
    int relativeDirection = 0;
    boolean moveSuccessful = true;
    java.util.Scanner sc = new Scanner(System.in);

    while(moveSuccessful) {
      //
      // B4
      //
    }

    System.out.println("Game over!");
    System.out.println("Score: " + s.score());
  }
}
public class SnakePart {
  //
  // Instansvariabler, A2
  //

  //
  // Konstruktor med parameterar, A3
  //

  //
  // ParameterlÃ¶s konstruktor, A4
  //
  public int getX() {
    return this.x;
  }

  public int getY() {
    return this.y;
  }

  public Direction getDirection() {
    return this.direction;
  }

  // Metod toString, A5

  //
  // Metod testPosition, A7
  //

  //
  // Metod move, A8
  //

  //
  // Metod isInside, A9
  //

  //
  // Mainmetod, A6
  //
}
```