CS341 Lab : GameObject and Ball moving along the x-axis

This lab practice will utilize an abstract class to describe the generalities of a Game Object.

Goals

1. Create an abstract class to represent a GameObject
2. Construct a Ball class that is implemented as a GameObject
3. Create a Canvas where movement will occur.
4. Create a Direction class to store values representing Directions.

Experiment 1: Ball object moves horizontally back and forth across the Canvas.

Diagram

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Application

Structure

Direction.java

**public** **class** Direction {

Provides a unique ID for the direction a GameObject can move.

**public** **static** **final** **int** ***NONE*** = 0;

**public** **static** **final** **int** ***UP*** = 1;

**public** **static** **final** **int** ***LEFT*** = 2;

**public** **static** **final** **int** ***DOWN*** = 3;

**public** **static** **final** **int** ***RIGHT*** = 4;

}

GameObject

**import** java.awt.Component;

Each GameObject has an x,y location, velocity, and a direction.

**import** java.awt.Graphics;

**import** javax.swing.Icon;

**public** **abstract** **class** GameObject {

**private** **int** x;

**private** **int** y;

Each GameObject has an image.

**private** **int** velocity;

**private** **int** direction;

**protected** Icon myImage;

**public** GameObject(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

velocity = 0;

}

**public** **void** draw(Component c, Graphics g) {

myImage.paintIcon(c, g, x, y);

}

// SETTERS AND GETTERS

**public** **int** getX() { **return** x;}

**public** **void** setX(**int** x) { **this**.x = x;}

**public** **int** getY() { **return** y;}

**public** **void** setY(**int** y) { **this**.y = y; }

**public** **void** setVelocity(**int** velocity) { **this**.velocity = velocity;}

**public** **int** getVelocity() { **return** velocity; }

**public** **int** getDirection() { **return** direction; }

**public** **void** setDirection(**int** direction) { **this**.direction = direction; }

//ABSTRACT METHOD

**public** **abstract** **void** move(Canvas c);

}

Each GameObject is moveable. move() is called by the Canvas and is performed in a gameloop.

Canvas

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.Graphics;

**import** java.util.LinkedList;

**import** java.util.List;

**import** javax.swing.JComponent;

**import** javax.swing.JFrame;

**import** javax.swing.Timer;

**public** **class** Canvas **extends** JComponent **implements** ActionListener{

// DEFAULT SERIAL NUMBER

Controls the animation.

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JFrame frame;

**private** Timer gameLoopTimer;

**private** List<GameObject> gameObjectList;

**public** Canvas() {

// TASK 1: CREATE A LIST OF GAMEOBJECTS THAT WILL APPEAR ON THE CANVAS

gameObjectList = **new** LinkedList<GameObject>();

// TASK 2: CREATE A WINDOW FOR THE APPLICATION

Animation is a game loop.

frame = **new** JFrame("Animation Canvas");

frame.setSize(800, 800);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.add(**this**);

// TASK 3: CONSTRUCT A TIMER FOR GAME LOOP

gameLoopTimer = **new** Timer(25, **this**);

gameLoopTimer.start();

// TASK 4: MAKE THE WINDOW VISIBLE

frame.setVisible(**true**);

}

**public** **synchronized** **void** addGameObject(GameObject sprite) {

gameObjectList.add(sprite);

}

paint() draws the GameObjects

**public** **synchronized** **void** paint(Graphics g) {

**for** (GameObject s : gameObjectList) {

s.draw(**this**, g);

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Canvas must implement the inherited abstract method

// ActionListener.actionPerformed(ActionEvent)

**public** **synchronized** **void** actionPerformed(ActionEvent e) {

**for** (GameObject gameObject : gameObjectList) {

gameObject.move(**this**);

}

repaint();

}

repaint() redraws the GameObjects. This is done after each game loop.

}

Ball

**import** javax.swing.ImageIcon;

**public** **class** Ball **extends** GameObject {

**private** **int** imageWidth;

**public** Ball(**int** x, **int** y) {

**super**(x, y);

setVelocity(5);

myImage = **new** ImageIcon("images/Ball.png");

imageWidth = myImage.getIconWidth();

}

Implementation of the abstract method required by GameObject

**public** **void** move(Canvas c) {

**if** (getDirection() == Direction.***LEFT***) {

setX(getX() + getVelocity());

// CHECK FOR A COLLISION WITH RIGHT WALL

**if** (getX() + imageWidth > c.getSize().getWidth()) {

setX((**int**) (c.getSize().getWidth() - imageWidth));

setDirection(Direction.***RIGHT***);

}

} **else** {

// MOVE BALL RIGHT (OPPOSITE OF LEFT)

setX(getX() - getVelocity());

// CHECK FOR A COLLISION WITH LEFT WALL: 0

**if** (getX() < 0) {

setX(0);

setDirection(Direction.***LEFT***);

}

}

}

}

MyMain

**public** **class** MyMain {

**public** **static** **void** main(String[] args) {

// TASK 1: CREATE A CANVAS FOR ANIMATION

Canvas canvas = **new** Canvas();

canvas.requestFocus();

//EXPERIMENT 1: CREATE A BALL OBJECT AND ADD IT TO THE CANVAS.

Ball ball = **new** Ball(100, 100);

canvas.addGameObject(ball);

}

}

**Experiment 2:**

A grid of Ball objects (15 x 15) moves horizontally back and forth across the Canvas.

Modify MyMain.

MyMain

**public** **class** MyMain {

**public** **static** **void** main(String[] args) {

// TASK 1: CREATE A CANVAS FOR ANIMATION

Canvas canvas = **new** Canvas();

canvas.requestFocus();

// EXPERIMENT 2: CREATE A COLLECTION OF BALLS AND ADD TO CANVAS

for (int row = 1; row <= 15; row++) {

for (int col = 1; col <= 10; col++)

canvas.addGameObject(new Ball(row \* 65, col \* 65));

}

}

}

Shape, background pattern

Description automatically generated