OBJECT ORIENTED PROGRAMMING

FINAL PROJECT

Diagram

Description automatically generated

**Student Information:**

Leonardo Richie

**Class Information:**

Class: L2BC

Lecturer’s name:

Jude Martinez

**Binus University International**

**School of Computer Science**

**2022**

Brick Break Game

1. Program description

A picture containing shape

Description automatically generated

After studying OOP for this semester, I plan to create my second game. My first game that I made is using python, so I try to do it in Java now. After learning inheritance, polymorphism, and interfaces, I plan to use all of them into my codes. For the OOP Final project, I plan to create a Brick break game. To play this game the user needs to break all the bricks in the level they choose to win the game. If the ball falls out of the screen the user loses. At the beginning, I look up at the internet how to create screen in java for my game. Then I check how the controller works and mechanism. After doing that I try to find a unique way to do it and I create classes including creating different kind of levels. To make the graphics, I use Jframe and Jpanel to create a new screen and draw the objects into it. This Program contain 6 files. First the driver file called BrickBreak, GamePlay class that store the game mechanism and MapGenerator that store map list, map generator. The 3 last file contains the position class as the parent and ball1, ball2, ball3 as the child class.

This game contains 9 levels, 1 home page, and 1 setting page. The user can access the level page once they are inside the home page. Inside the setting page, the user can do some modifications, for example changing the number of balls, the speed of the ball, and the length of the paddle. This is what makes the brick game special and more interesting. I use Visual Studio Code for my project since I use it often.

1. Project Specifications

**Project Purpose:**

The code is to create a brick game which contain gameplay file which contain the mechanism of the game, MapGenerator file to generate map levels and a driver file called BrickBreak. To create it the project requires Jframe and Jpanel.

**Project Audience:**

People who want to learn more about classes and the brick game that use oop classes.

**Project Aim:**

Create a brick break game with different levels. To win this game the player need to break all of the brick inside the level they choose. There are 9 levels, they can also modify the levels by changing some condition for example like the number of balls.

**Project Requirements:**

* Gameplay file to store the gameplay mechanism, ball classes, paddle.
* Jframe and Jpanel to create a page and draw objects into the screen.
* MapGenerator file that stores level maps and generate it.
* Driver file
* Packages to support the game (KeyEvent, KeyListener, ActionEvent, ActionListener, Timer, Jpanel, color, Graphics, Rectangle, Graphics2D, Audio System.
* Inheritance, Polymorphism, Interfaces.

1. Class DiagramDiagram

   Description automatically generated
2. Variables

Text

Description automatically generated

Stating variables are needed when making the game. Inside the gameplay class, I stated some variables that are needed inside the game. To state each variable, data types also need to be typed. There are 3 boolean data types of variables which is play, gameplayOn and frontPage. The play variable is to state whether the gameplay in each level is still running. If the gameplay has stopped it will be false and the user will need to choose another level or reset that level. GameplayOn variable is similar to the play variable, but it is used to stop the paddle from moving when the user loses. The totalworldBricks variable is used to declare how many bricks in every different level maps. Totalbricks variable are used to state the number of bricks which later will be used to finish the game once there are no bricks left in the level. The timer and delay variables are used for the ball movement, paddle movement and animation. Delay is equal to 8, the longer the delay the slower the animation. Next will be the player position variable which is playerX and playerY. Both are the coordinates of the player position from x and y. WorldspeedX and worldspeedY are used to declare the speed of the ball inside the game. They can be changed through the settings. The worldNum variable are used inside the game to pick which level map will be shown in each level. Since the game has 2 page and 9 levels. The 2 pages use page variable so that the user can move from 1 page to another.

I have created objects for each ball class which is drawn to the screen when the player is inside the level. The numberofBall variable is to state the number of ball (set 1 as default). And next is the map generator class map, it creates the mapGenerator object to access map and modify it.

Text

Description automatically generated

There are some variables stated in the mapStats class. The int x are used to state the number index in each array to access the level. The brickWidth and height are to declare how long the width and height for each brick. The numBricks variable are used to state how many bricks are there in each level. To store the map, multidimentional array is used to store each level of the map in 1 array.

1. Setting up gameplay () Method

A screenshot of a computer

Description automatically generated with medium confidence

So, this is the constructor method for the Gameplay class. When the Gameplay object is created it directly generate a map with a worldnum 9 which means that there is no brick on the screen. It also adds keylistener to the object. setFocusable True and setFocusTransversalKeysEnabled false is to get the input for each key pressed by the user. Next is the timer, new Timer with delay variable inside it and this means that to create a timer with the delay 8 (stated in variable). Timer.start() is to start the timer as long as the program runs.

1. Scores and resetting when the player lose

**Scores**

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

Description automatically generated

Text

Description automatically generated

Here is how the score is coded. Worldnum 9 is an empty bricks map which is used for the front and setting page so the level will only appears when the user are inside one of the levels (worldnum 1-9). First set the color of the font to white, set font (setting up the name of font, bold and size). Next will be drawing the string to the screen

The x and y are the coordinates of the string which is 590 and 30 starting from zero in the left top corner. The score variable will be added every time a ball hits a brick starting from 0.

**Resetting**

A screenshot of a computer

Description automatically generated with medium confidence

As the user loses (all the balls fall out from the screen) the play variable will be false. To reset the game, we can press enter key to reset and generate the same map with tits bricks reset. With the resetGame() function it will reset all the paddle and ball stats. And the repaint() function is to redraw all of the component into the screen similar to updating it.

Text

Description automatically generatedText

Description automatically generated

This is the reset function. This resetGame() is used to reset all the stats inside the game.IT resets the ball direction,ball position. Player position and also the score. It also returns the gameplayON to true. When gameplayON is false the paddle cannot be moved. So when it is back become true the user can move the paddle again. The ball will be reset to its original position. Each direction of the ball is also set depending on the worldSpeed (set by the player in the setting or by default). TotalBricks it to reset the number of bricks before the ball breaks (each level has different total bricks). And last is reset the player position in the middle and its score to zero.

**When the player finish breaking all the Bricks.**

Text

Description automatically generated

When the totalBricks is 0 (no bricks left inside the level), the player will be in the front page for the user to choose another level. It use the mapgenerator on index 9 which is the empty map with no bricks for the page.In Mapgenerator(10,10,9), 10 10 means that the size of the brick map is 10 bricks by 10 bricks. The worldnum will be 9. Gameplay and game will be false since the user is not playing the level.

1. Movement

A picture containing text, silhouette

Description automatically generated

**Paddle**

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

To move the paddle, this program uses keyboard arrow keys that’s why we use the keyEvent.VK\_KEYS and getkeyCode() to get input of any keys inside the keyboard. The paddle can move in four directions which is up, down, left, and right. The if page ==2 statement is to run when the player is inside one of each level so the paddle will not move when the player is in the main or setting page.

The VK\_RIGHT keys have 2 functions. First is to move the player from the main page to the setting and the second is to move the paddle to the right. VK\_LEFT is to move the paddle to the left. VK\_TOP to move the paddle up and VK\_DOWN to move the paddle down. Each movement has an if statement which is to prevent the paddle from moving out of the screen. The paddle can move between 10 and 600 in x direction and 475 and 500 in y direction. So, when it moves more than the position, it will not move again and hold in that position. The else statement is to show that if it is not more than the range then the paddle will move up, down, left and right depending on the key pressed by the user.

**Ball**

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated

First the position class is the parent class for all ball classes created. It has attributes for ballposX,ballposY, ballXdir, ballYdir. Ballpos is to state the position of the ball in x and y coordinates. Balldir is to state the direction of the ball. Plus, and minus integer determine the direction of the ball. There are some methods, inside the position class. posX and posY are similar to get function which to return the position of the ball in x and y. resetPosition() is to set the position to zero for both ball. addX and addY is to add integer to the position of the ball in x and y. It is used to move the ball position. Next is the set function to set the position for the ball. dirX and dirY are the get function for the ball direction that return the direction of the ball in x and y. FlipXdir and FlipYdir is to flip the object direction by multiplying it with -1. setDir() is to set the direction of the ball and resetDir to reset the direction of the ball to zero.

A screenshot of a computer

Description automatically generated with medium confidence

This is the child class that extends from the parent class. Ball1, ball2, ball3 have different resetposition method, by using the polymorphism. So that every position for each ball are different

A screenshot of a computer

Description automatically generated with medium confidence

Both ball and paddle will be always updated as the game runs by using the fillOval and fillRect with its position updated in x and y using the ball.posX() and ball.posY().

1. Drawing Objects into the screenGraphical user interface, text, website

   Description automatically generatedA screenshot of a computer

   Description automatically generated with medium confidenceText

   Description automatically generated

To draw object into the screen, fill function is needed, fillOval and fillRect. setColor is used to set the color for each shapes. FillOval is to draw circle object into the screen and fillRect is to draw rectangle to the screen. Inside the bracket, there are x and y to state the position and its width and height. As for the ball, I use the ball.posX() and ball.posY() function for each ball to get the position of x and y stored in the class (return position of x and y). First, Background is created with its width and height as the size of the screen and the color is black. Next is drawing map into the screen by using the map.draw((Graphics2D)g). More details about drawing map will be explained in the map generator part. When drawing the border, there are three rectangle shapes created for the border. The position for each is in the top, left and right. Drawing the ball and paddle are similar, only in different shapes (rectangle for paddle, circle for ball). Both are inside the if function which makes them to be drawn only when the player is playing inside levels. There are 3 balls, so it depends on the game. If it is set to 2 balls, there will be 2 balls drawn.

1. Main Driver

Text, chat or text message

Description automatically generated

This is the driver where the code runs, the name of the file is BrickBreak. First is to create the gameboard using jframe and create the gameplay class object which contains all the mechanism inside the game. setBounds is used to set the size of the jframe object (700x600). Next is to setTitle which is set the title of the app on the top corner left. I don’t set the game to resizable so it will be false. The game need to appear so setVisible will be true.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE) is for the user to closed the game when they press the x button on top right. And last is to add gameplay into the frame to put all the mechanism inside it. The Playsound(Gamesound) is to play the sound effect of the game so it will run throughout the game.

1. Map Generator

**MapStats class**

Graphical user interface, text

Description automatically generated

Inside the MapGenerator file there is also another class that store attributes for the MapGenerator which contains the integer x to identify which level are going to be picked, the width and height of brick, number of bricks and a multidimensional array that contain the brick location

**MapGenerator class**

Text

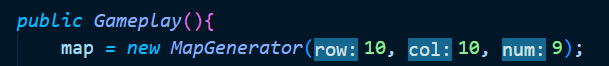
Description automatically generated

The mapGenerator contains method that create the map so that it will be drawn into the screen. It extends the MapStats class to take its attributes. There is a constructor method inside the class which is to count the brickwidth and height. The number of columns and rows I used for the game is 10x10, so each brick width is 54 and height is 35. The draw function is to set the color and draw each rectangle into the screen which work as bricks in the level. I used nested loop to access the array. The array consists of 1 and 0 numbers. It means that every number 1 contain bricks and 0 is empty. Inside the loop there is also if statement to show that every 1 in the array, create a brick by setting its color to red and fill the rectangle with the location of j \* brickWidth + 80 and I \* brickHeight + 50. +80 and +50 is to locate the bricks in the middle not at the corner of the screen. And last the setBrickValue method is used to change the value inside the array depending on the location by using int row and int col. So, the main function of this method is to change the number 1 in the array to zero when a ball hit one of the bricks so it will disappear.

Graphical user interface, text, website

Description automatically generated

Graphics2D is a class that extend Graphics class to provide more control to geometry, changing positions and it is mostly used for rendering 2d dimensional shapes, text, and images.



Inside the gameplay class, a MapGenerator object is created depending on the page. As the front page it will be empty without bricks so the mapGenerator will take the index 9 from the array which contains empty brick.

A screenshot of a computer

Description automatically generated with medium confidence

For each levels the mapGenerator is also updated into the variable depending on the level. Level 1 uses index 0 of the array map and level 2 uses the index 1 until level 9.

1. Collision

Logo

Description automatically generated with medium confidence

Collision is the main solution to this brick game. It is used to remove the bricks once the ball hit the brick and bounce the ball in the reverse direction. Not only that it also helps to bounce the ball back between the paddle and the wall.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

To make this happens, Inside the gameplay file there is an override method which is a method inside the ActionListener interface. The override is to gain access to the function from actionListener to this gameplay class. This method is called after the user perform and action. First there is an if statement, so it only works when the play is true which means only when the user is playing one level. The 3 if statement of rectangle is to show that once the ball intersects with the paddle it will bounce back in the opposite direction it hits. To bounce the ball back, I use the flipYdir() and flipXdir() function from the position class to flip the direction of the ball. The position for both keeps updating with the variable position for each object. -2 is also added to the ball position so that the ball does not crash with the paddle when both are moving and collided (ball move 2 positions up). Next is the nested loop which is to access each brick that are in the array. It also creates the brick x and y position depending on the width and height and put the brickWidth and Height into a new variable. Next is creating the rectangle object for each ball and bricks, getting its position in x and y using class function and its width and height.

The next if statement (second Picture) is to run when the ball intersects with one of the brick. Map.setBrickValue(0,I,j). I and J is the column and rows, so it is set to 10. The setBrickValue is a method inside the MapGenerator class. So, when the ball hits a brick, it will set the value to 0 (the brick will disappear). The total bricks are also minus 1 because 1 brick disappears and the score adds 5. After hitting a brick, the ball will bounce back into the opposite direction. So the if statement shows that the ball can come from the left and right. The intersects function is to show if the 2 items are intersecting with each other. Ball1.posX()+19 <= brickRect.x (ball is coming from the left of the brick) or ball1posX() + 1 >= brickRect.x + brickRect.width (ball coming from the right) will oppositely change direction on x using the flip method from the position class. and if they don’t come from left and right, that means the ball will come from top or bottom so it will change direction on y. After changing direction, the loop will break because a brick can only be hit one time. There are 3 if statement intersection which is between first ball with bricks, second ball with bricks and third ball with bricks if there are more than one ball inside the level.

1. Music

Text

Description automatically generated

To create music for the game, 3 imports are needed to read files, open, and put it into clip object.



Graphical user interface, text

Description automatically generated

Logo

Description automatically generated

First is to import the game.wav file into the GameSound object. The Playsound function is to play the music. First with the Clip object, getClip is to play the sound again and getAudioInputStream is to get the input from which file do the user want to play. Start() is to start playing the clip and loop continuously until the program ends. The try and exception is to show that the function playSound requires a sound file, if there is no file it will do an exception. But I leave it empty because the name of the sound is already named and placed in the file so there will be no worry about error when running in the game. And last is using the method PlaySound to the GameSound that contain Game.Wav (the music file for the game).

1. Page managing

Text

Description automatically generated Text

Description automatically generated A picture containing graphical user interface

Description automatically generated

There are 3 pages inside the game. To change between pages, I use the page variables which stores number from 0-2 depending on which page. 0 as in the level page, 1 is the front page and 2 for the setting page. And to change pages, if statement is used in every command.



Because it starts from the main page cover, the page variable will be equal to one.

Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

If the page is 1, that means front page is true (does not show paddle and ball). setFont and drawstring are the main code to create main page and setting pages. setColor is to set the color of the font and drawstring is to draw it into the screen. X and y are the position of the text based on x and y coordinates. If page is zero, which means it is the level page. It will create bricks and ball. The ball depends on how many balls are there in the level. By using the setcolor, fillOval and fillRect, ball and paddle can be created.

Text

Description automatically generated

To go to the setting page, the user needs to press right arrow in their keyboard to move to the setting page and to go back press the left keyboard. By using the if statement in each key, the page will change. If page is 1 which mean it is in the main page, when the user press right, it will go to page 2. The same thing will go to the setting page, when the user press left, it will go to page 1. The front page will become false once the user is not in the main page.

A screenshot of a computer

Description automatically generated with medium confidence

As for the level page the user can press any number from 1-9 on the main page. With the if function it will create the level world with a total bricks (depending on each level), page 0 (means the level page), resetGame() (set all the position of ball and paddle), create a new map and set world num to zero since it is level 1. The repaint() is used to update the screen. The if statement is to show that play is false (the game has not started yet) and frontPage == true which means that the user is in the main page, because if they are inside the setting page they cannot go to levels.

1. Customization

Text

Description automatically generated

There are 3 parts that can be customized inside the game. Ball speed which is to make the ball faster, Paddle size which can be set to long, normal and short, and also the number of ball with a maximum 3 ball.

A screenshot of a computer

Description automatically generated with medium confidence

To change those 3 parts, every string related to the 3 parts have variable that will change automatically depending on the user. To change the variables, the user can use keyboard keys.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

This is how the keyboard key works. Each of the keys can only be press if the page == 2 which means the user is in the setting page (as shown on the code in if statement). To change ball speed the player can press G or F depending on how fast they want. G is for 1x which is the normal speed and F is for 2x speed which is more faster. It will directly change the worldSpeedX and worldspeedY 2 times more faster. For the width of the paddle, the user can press N, S, E keyword to change. N is for normal, S is for short and E is for Extreme(super short). It will change the paddle size to 100, 50, 25 and change the nameLength variable which is used to change the word in the setting. To change the number of ball the user can directly press the number key of the keyboard 1, 2, 3. It will directly change the numberofBall variable which is used in the setting as a word and change the number of ball inside each level once played.

1. Resource

Github link: <https://github.com/LeonardoRichie/JavaSirJudeFinalProject>

Reference:

https://www.youtube.com/watch?v=K9qMm3JbOH0&t=2930s

<https://stackoverflow.com/>

<https://javapointers.com/>

https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html