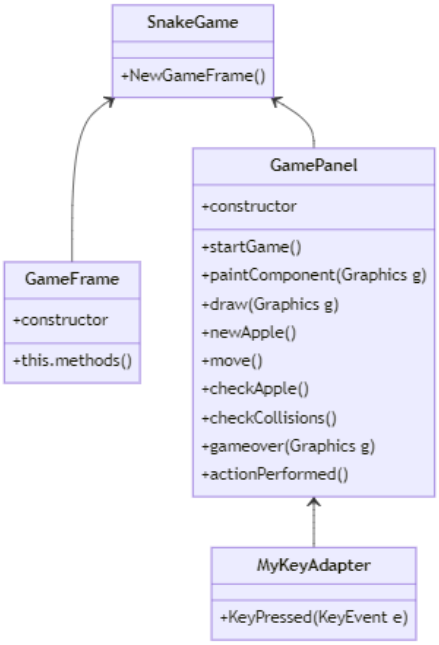
IMPLEMENTATION MANUAL



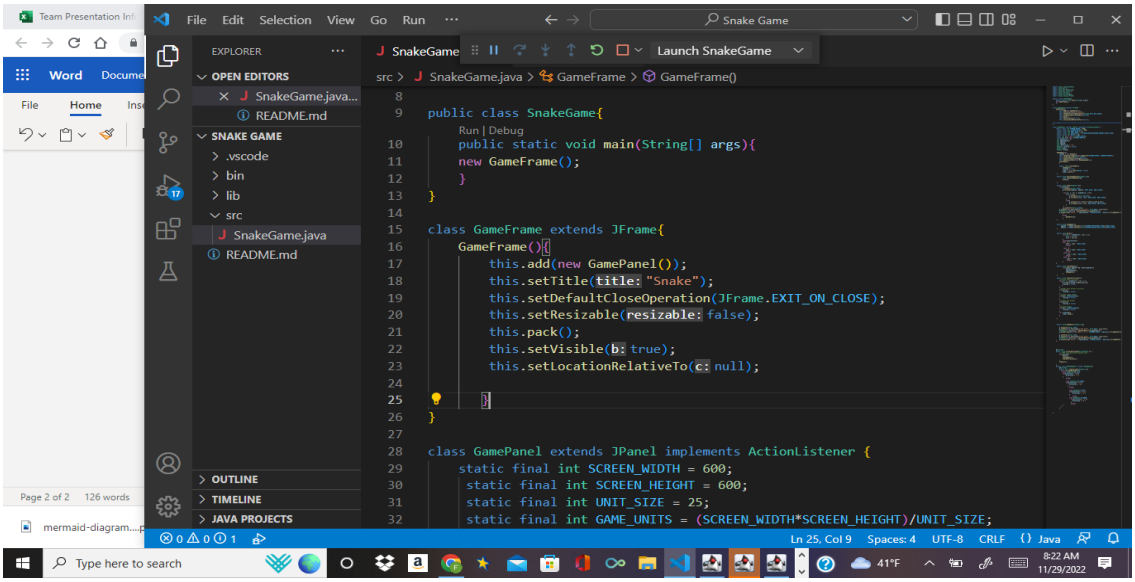
The diagram above shows the basic and the key structure of the whole program. It has three different classes with one additional interclass called “MyKeyAdapter.”

*Staring with the class that holds the main method called “SnakeGame,” has one method called “new GameFrame()” which is the shortcut form of the instance “GameFrame.”

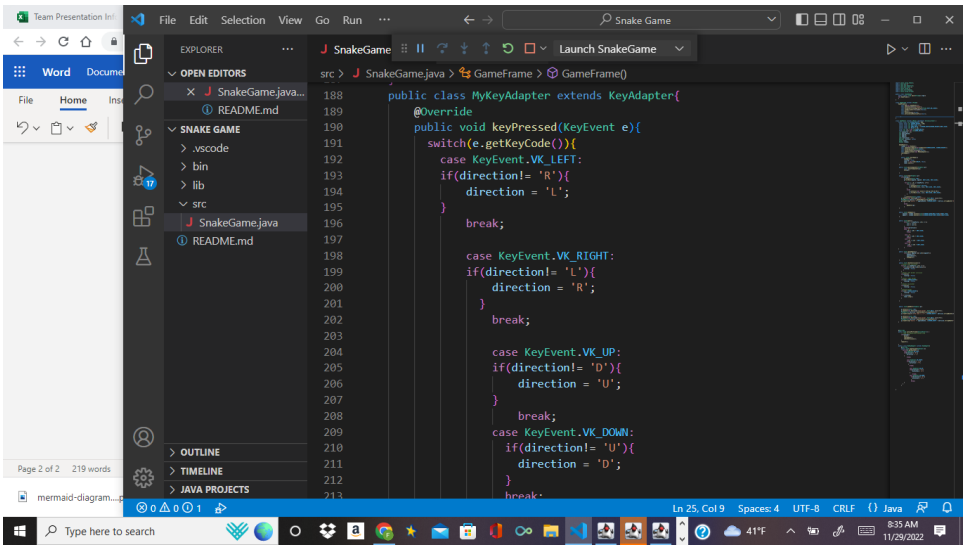
* The second class is called “GameFrame” which is in charge of creating the key frame of the game. It extends JFrame, which is a top-level container that provides a window on the screen. This class only holds the construction of the frame. With “this. (methods)” the frame can be constructed as desired, such as for setting the

title, “this.setTitle(“Snake”)” is used, and to make the frame visible

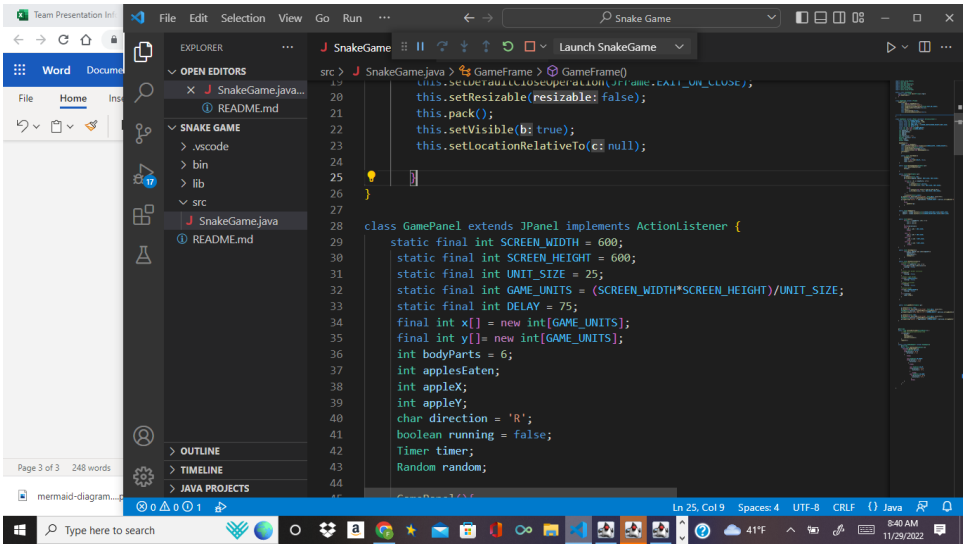
“`this.setVisible(true);” is used etc.`



*Everything else regarding the building will be handled on the “GamePanel” class, which extends JPanel, which is a part of the Java Swing package, a container that can store a group of components. Then it implements ActionListener interface, which handles all action events when the user clicks on a component. It has an Overridden unimplemented method to perform the actions as the users want. The GamePanel class has an interclass called “MyKeyAdapter” which extends KeyAdapter, it controls and performs the keys/arrow/direction in order to command the snake object to move with switch statements by the method called “KeyPressed”.



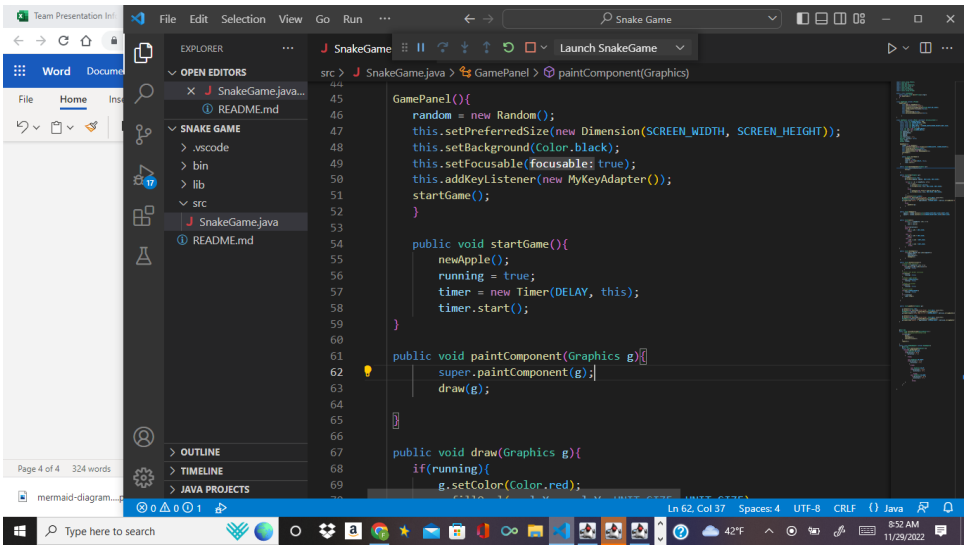
*Before constructing the “GamePanel” itself, some things need to be declared or initialized such as the dimensions of the panel, unit size, timer, bodyparts of the objects and X and Y co-ordinates of the objects as they move etc.

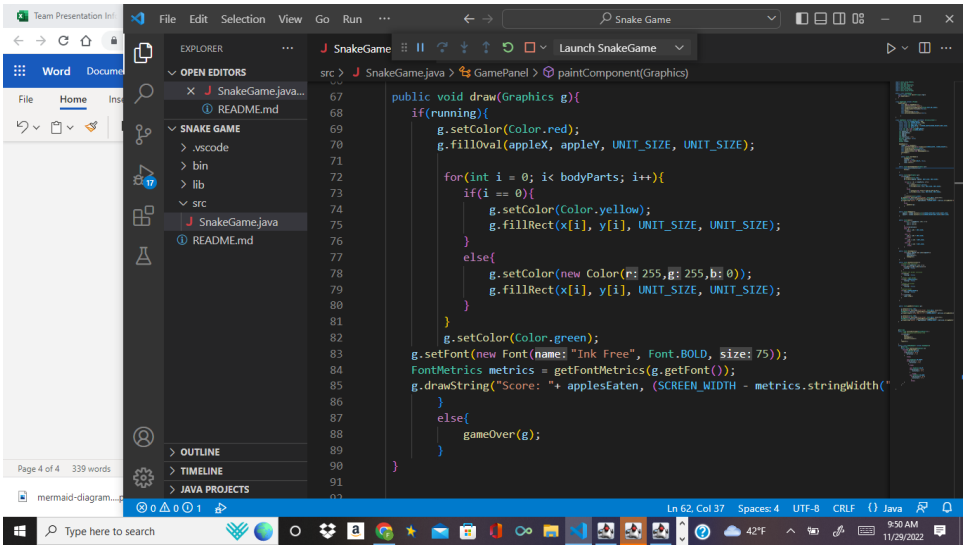


*Within the constructor, after importing required components, “this.” methods are used as desired to create dimension and background, to make it focusable enough

while running. Then essential components to run the game successfully, such as “KeyListener” is added and startGame(); method is called.

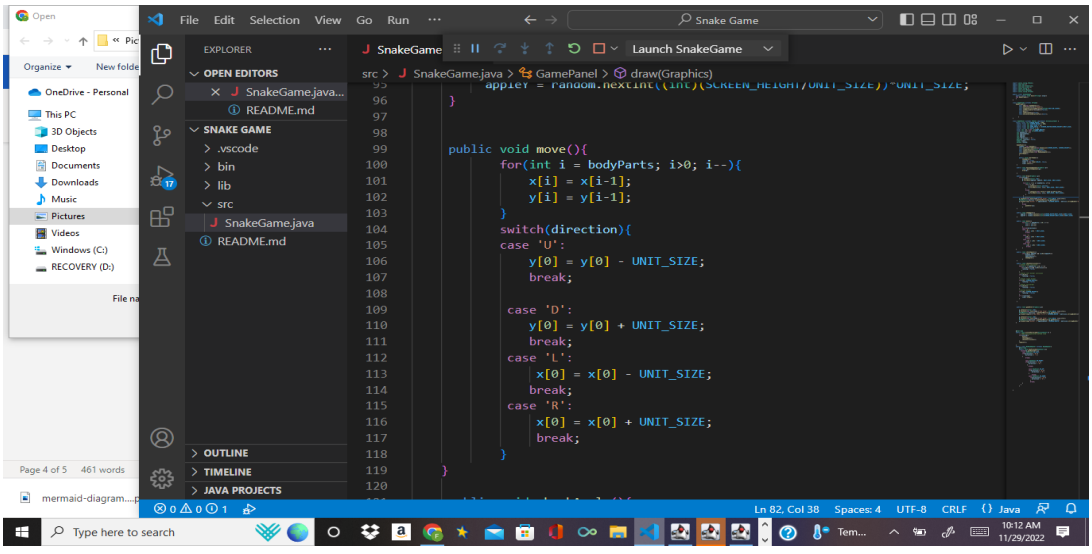
* From here, it’s all about creating methods to control and design the game fully now. Starting with “startGame” method, the appearance and location of the apple is controlled by the importation of Random. Boolean “running” is set to “true” along with the timer to start the game.

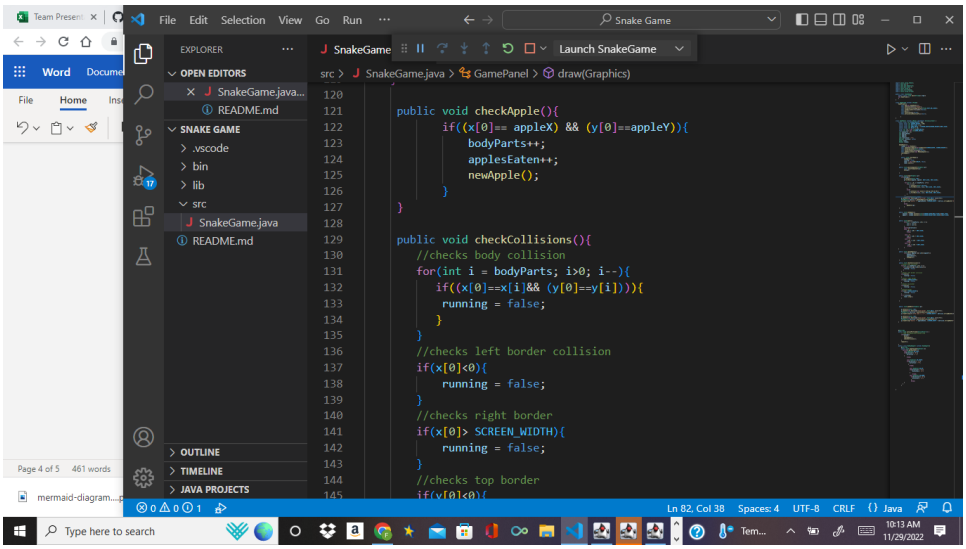


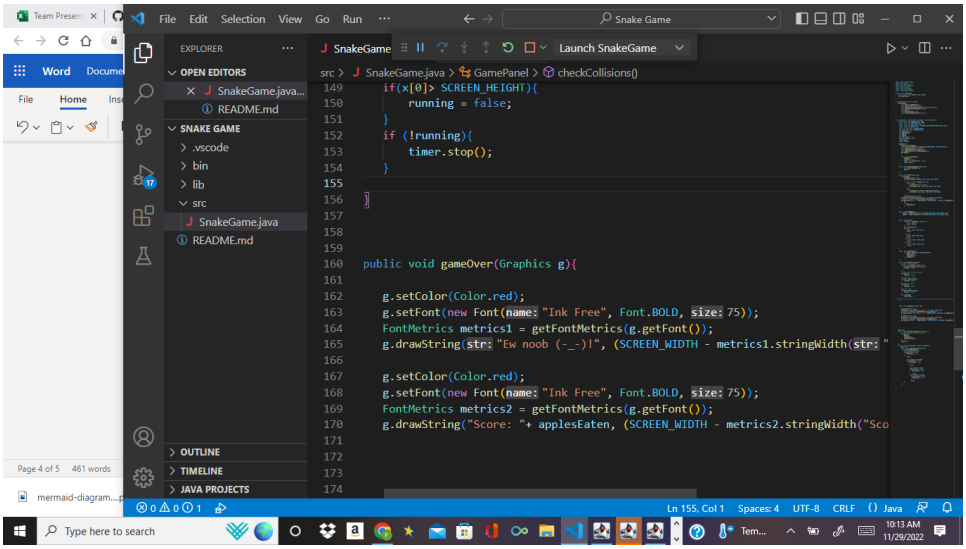


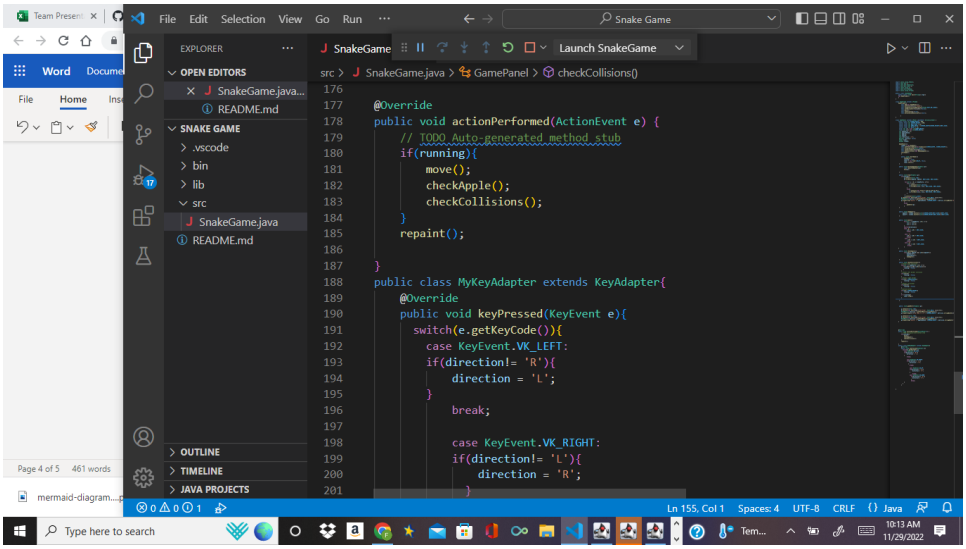
*The draw and paintComponent methods set the visual of the objects (apple and snake). By set and fill methods, body parts and apples are colored as desired by using RGV values and coordinates are placed accordingly. RGV values of specific colors can be googled and used to your liking. The whole draw method is surrounded by if statement while running at constant speed, or else, it stops drawing when the game is over, and the statement is terminated.

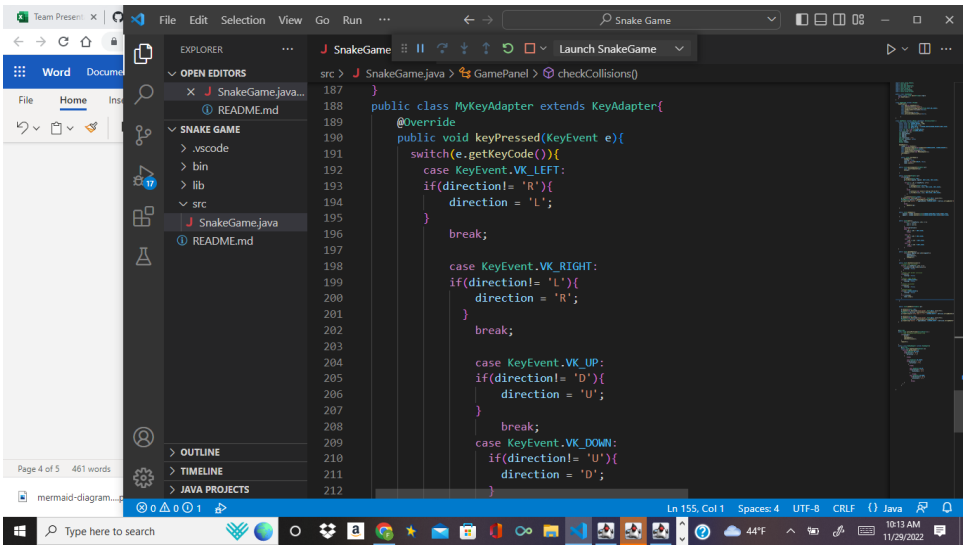
* The move method includes a for loop to iterate bodyparts through decrementing by one so that it keeps shifting one unit as the switch statements run by given direction as the user pleases. Array x and y are set according to the movements of their coordinates. Lastly, as methods are performed one by one in order, for the gameOver method, choose a string of texts as your liking by using the given codes below, here “Ew Noob!” is used with the display of earned score along with it when the game is over.

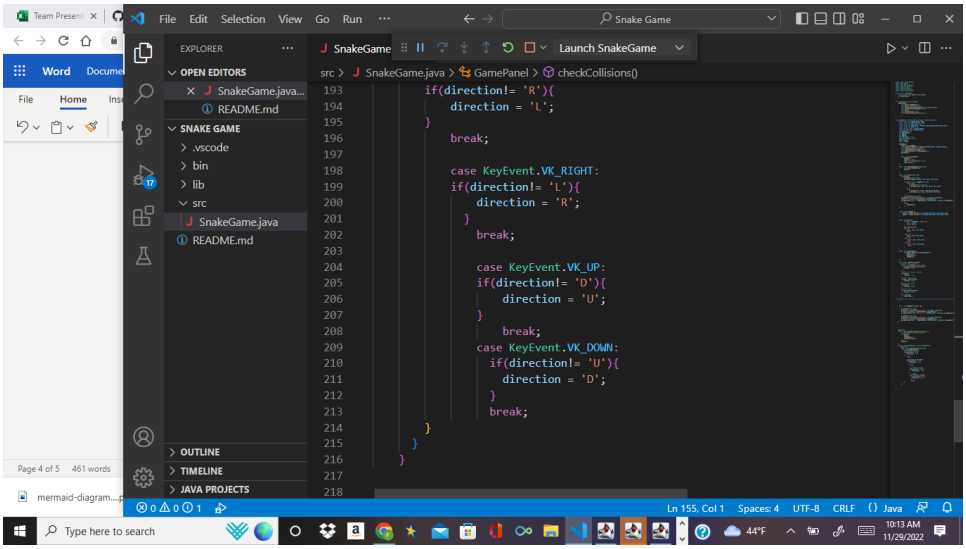



File Edit Selection View Go Run
Snake Game
Launch SnakeGame
EXPLORER
OPEN EDITORS
SnakeGame.java...
README.md
SNAKE GAME
.vscode
bin
lib
src
SnakeGame.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
src > SnakeGame.java > GamePanel > draw(Graphics)
public void checkApple(){
if((x[0]== appleX) && (y[0]==appleY)){
bodyParts++;
applesEaten++;
newApple();
}
}
public void checkCollisions(){
//checks body collision
for(int i = bodyParts; i>0; i--){
if((x[0]==x[i]&& (y[0]==y[i]))){
running = false;
}
}
//checks left border collision
if(x[0]<0){
running = false;
}
//checks right border
if(x[0]> SCREEN_WIDTH){
running = false;
}
//checks top border
Ln 82, Col 38 Spaces: 4 UTF-8 CRLF Java
Type here to search
10:13 AM
11/29/2022


File Edit Selection View Go Run
Snake Game
Launch SnakeGame
EXPLORER
OPEN EDITORS
SnakeGame.java...
README.md
SNAKE GAME
.vscode
bin
lib
src
SnakeGame.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
src > SnakeGame.java > GamePanel > checkCollisions()
if(x[0]> SCREEN_HEIGHT){
running = false;
}
if (!running){
timer.stop();
}
}
public void gameOver(Graphics g){
g.setColor(Color.red);
g.setFont(new Font(name: "Ink Free", Font.BOLD, size: 75));
FontMetrics metrics1 = getFontMetrics(g.getFont());
g.drawString(str: "Ew noob (-_-)!", (SCREEN_WIDTH - metrics1.stringWidth(str:
g.setColor(Color.red);
g.setFont(new Font(name: "Ink Free", Font.BOLD, size: 75));
FontMetrics metrics2 = getFontMetrics(g.getFont());
g.drawString("Score: "+ applesEaten, (SCREEN_WIDTH - metrics2.stringWidth("Sco
Ln 155, Col 1 Spaces: 4 UTF-8 CRLF Java
Type here to search
10:13 AM
11/29/2022


File Edit Selection View Go Run
Snake Game
Launch SnakeGame
EXPLORER
OPEN EDITORS
SnakeGame.java...
README.md
SNAKE GAME
.vscode
bin
lib
src
SnakeGame.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
src > SnakeGame.java > GamePanel > checkCollisions()
@Override
public void actionPerformed(ActionEvent e) {
// TODO Auto-generated method stub
if(running){
move();
checkApple();
checkCollisions();
}
repaint();
}
public class MyKeyAdapter extends KeyAdapter{
@Override
public void keyPressed(KeyEvent e){
switch(e.getKeyCode()){
case KeyEvent.VK_LEFT:
if(direction!= 'R'){
direction = 'L';
}
break;
case KeyEvent.VK_RIGHT:
if(direction!= 'L'){
direction = 'R';
}
Ln 155, Col 1 Spaces: 4 UTF-8 CRLF Java
Type here to search
10:13 AM
11/29/2022


File Edit Selection View Go Run
Snake Game
Launch SnakeGame
EXPLORER
OPEN EDITORS
SnakeGame.java...
README.md
SNAKE GAME
.vscode
bin
lib
src
SnakeGame.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
src > SnakeGame.java > GamePanel > checkCollisions()
}
public class MyKeyAdapter extends KeyAdapter{
@Override
public void keyPressed(KeyEvent e){
switch(e.getKeyCode()){
case KeyEvent.VK_LEFT:
if(direction!= 'R'){
direction = 'L';
}
break;
case KeyEvent.VK_RIGHT:
if(direction!= 'L'){
direction = 'R';
}
break;
case KeyEvent.VK_UP:
if(direction!= 'D'){
direction = 'U';
}
break;
case KeyEvent.VK_DOWN:
if(direction!= 'U'){
direction = 'D';
Ln 155, Col 1 Spaces: 4 UTF-8 CRLF Java
Type here to search
44°F
10:13 AM
11/29/2022


File Edit Selection View Go Run
Snake Game
Launch SnakeGame
EXPLORER
OPEN EDITORS
SnakeGame.java...
README.md
SNAKE GAME
.vscode
bin
lib
src
SnakeGame.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
src > SnakeGame.java > GamePanel > checkCollisions()
if(direction!= 'R'){
direction = 'L';
}
break;
case KeyEvent.VK_RIGHT:
if(direction!= 'L'){
direction = 'R';
}
break;
case KeyEvent.VK_UP:
if(direction!= 'D'){
direction = 'U';
}
break;
case KeyEvent.VK_DOWN:
if(direction!= 'U'){
direction = 'D';
}
break;
}
}
Ln 155, Col 1 Spaces: 4 UTF-8 CRLF Java
Type here to search
44°F
10:13 AM
11/29/2022