Hasif Ahmed, Wenting Li, Kerwin Chen

Hidetada UML Diagram Period 1

Game

+ Player[] = new Player[4]/

void setup()
- create size of panel
- instantiate 4 players
void draw()
- constant refresh of screen
- backrgound, player movement, and filling of backrgound with shapes
void keyPressed()
-controls movement of players and bomb deployment

Player

float xcor,ycor (current location of player)
float xorig,yorig (spawn location)
final static int ALIVE = 0 (state of player)
final static int DEAD = 1 (state of player)
Bomb x (bomb that player deploys)
int bombs (number of bombs)
int lives = 3
int state (variable to hold state of player)
color c (color of player)

public Player(float x, float y)
- overloaded constructor
- create player at designated location
int getLife()
- return number of lives
void update()
- location of playr
void check(Player[] players)
- check if each player is within the range of a bomb explosion
- if so subtract lives, respawn
void die()
- game over
void dropbomb()
- generate bomb by players coordinates

Bomb

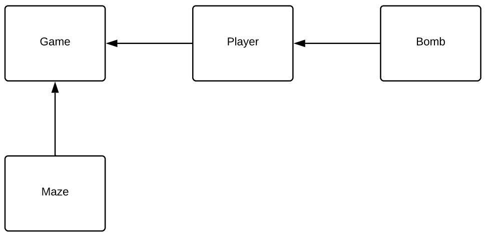
float xcor,ycor (current location of bomb)
final static int A = 0 (state of bomb)
final static int B = 1 (state of bomb)
float state (variable to hold state)
color c (color of bomb)
boolean dead = false (if bomb and explosion are gone then true)
boolean explosion = false (if bomb is in exploding state then true)

public Bomb(float x, float y)
- generate bomb at x and y coordinates
void update()
Stage 1- Bomb is deployed
Stage 2- Bomb explodes
boolean getDead()
-return value of dead
boolean getExplosion()
- return value of explosion

Maze

-not any as of yet

public Maze()
-generate maze algorithm
-implement onto background of panel


Game
Player
Bomb
Maze