# LAPORAN PROGRAM 2048

UWIKA


Nama Anggota :

Michael Wilbert G 311.17.002

Matthew Anastasius 311.17.019

# BAB I
# PENDAHULUAN

## A. Latar Belakang

Pada umunya kita semua pasti sudah mengetahui permainan yang bgernama " 2048 " dan permainan tersebut sangat bermanfaat bagi kita untuk mengasah pola pikir kita untuk dapat menyusun strategi. Maka dari itu permainan sangat digemari bagi kaum anak-anak muda, karena didalam permainan tersebut diundang para player untuk memecahkan / menyusun angka tersebut sampai mencapai angka dari nama permainan tersebut yaitu " 2048 ".

## B. Alur Permainan

1. Pertama-tama user / player akan membuka game " 2048 ".
2. Di dalam Window yang muncul akan ditampulkan grid 4x4 dan 2 nomor random di posisi yang random juga dalam grid, 2 nomor tersebut hanya boleh 2 / 4 saja.
3. Setiap ada perubahan (digerakan), maka board akan menggerakan semua nomor ke arah tersebut (Menggunakan arrow atas, kiri , kanan,dan bawah), dan jika ada nomor yang sama maka akan ditambahkan, jika tidak hanya bergeser saja.
4. Setiap perubahan terjadi, maka akan mengespawn angka 2 / 4 di tempat yang value nya masih 0 secara random (posisi nya).
5. Game akan berakhir jika Board penuh dan sudah tidak bisa digerakan ke segala arah.
6. Setelah game berakhir maka akan muncul satu window untuk mereset permainan.
7. Game bisa dianggap selesai jika terbentuk angka 2048, tetapi tetap bisa dilanjutkan.

## C. Konsep Object Oriented Programming

1. Terdapat 3 *class* untuk menjalankan Program. (main, window, dan Game)
    1.1. Main bertugas untuk menjalankan program nya dengan memanggil window.
    1.2. Window bertugas untuk menampilkan program.
    1.3. Game bertugas sebagai pengatur *logic game* nya.

2. *Class* window terdapat constructor sebagai penginisialisasi program yang tugas nya mengeset gridlayout, size window , dan memasukan objek objek JLabel didalam sana. Update bertugas untuk mengupdate tampilan JLabel dengan menyesuaikan array dari objek Game sehingga tampilan sesuai dengan hasil logic di dalam game, selain itu terdapat keyListener pada window agar dapat menerima inputan dari keyboard yang kemudian

dapat memanggil fungsi dari game untuk menjalankan array yang kemudian akan di update dan disesuaikan dengan tampilan di Window.

3. Sedangkan di dalam *class* game, terdapat fungsi fungsi untuk menggerakan array , spawner angka , dan pengecek apakah masih ada kemungkinan gerakan di dalam array / tidak. Selain itu terdapat fungsi yang bertugas untuk mengkopi array asli , yang kemudian dapat sementara di jadikan array yang dapat kita rubah rubah dan di kembalikan jika tidak terjadi sesuatu hal yang diinginkan (jadi misal tak ada perpindahan maka akan dikembalikan disaat sebelum penambahan (mengkopi tabel copy)).

# BAB II

# PROGRAM

## - Bagian Bagian Program :

* Class :

- Penjelasan :

*class* adalah merupakan suatu *template object* yang dapat kita perumpamakan sebagai *blueprint* / rancang bangun suatu *object* , yang dimana memiliki variabel variabel , dan fungsi fungsi di dalamnya.

- Bagian Program yang menggunakan Class :

utama.java

```
1
2   public class utama{
        Run | Debug
3       public static void main(String args[]){
4           new window();
5       }
6
7   }
```

window.java

```
17  import javax.swing.SpringLayout.Constraints;
18
19  public class window extends JFrame implements KeyListener{
20      private int     height  = 600;
21      private int     width   = 600;
22      private String  title   = "2048";
23      Game    board           = new Game();
24      JLabel  boardTile[][]   = new JLabel[4][4];
25      Border  border          = BorderFactory.createLineBorder(Color.BLUE,5);
26
```

Game.java

```
1   import java.util.Random;
2
3   public class Game{
4       private int boardLabel[][] = {
5           {0 , 0 , 0 , 0 },   // [0][0] , [0][1] , [0][2] , [0][3] ,
6           {0 , 0 , 0 , 0 },   // [1][0] , [1][1] , [1][2] , [1][3] ,
7           {0 , 0 , 0 , 0 },   // [2][0] , [2][1] , [2][2] , [2][3] ,
8           {0 , 0 , 0 , 0 }    // [3][0] , [3][1] , [3][2] , [3][3] ,
9       };
```

* Abstract Class :

- Penjelasan :

*Abstract class* adalah *class* yang yang lebih fokus kepada suatu objek saja , dan fungsi fungsi nya tidak perlu di isi , dan biasanya hanya dibuatkan nama fungsi yang bisa di lakukan kemudian class mana pun yang men *extends* class abstract diwajibkan meng override fungsi tersebut.

- Bagian Program yang menggunakan Enkapsulasi :

Tidak ada , karena kita hanya menggunakan 1 Window dan 1 Class sebagai gamenya.

* Enkapsulasi :

- Penjelasan :

Tidak semua data dapat langsung diakses variabel nya secara lagsung dari class lain.

- Bagian Program yang menggunakan Enkapsulasi :

```
public int[][] getBoardLabel(){
    return this.boardLabel;
}
```

```
private int boardLabel[][] = {
    {0 , 0 , 0 , 0 },   // [0]
    {0 , 0 , 0 , 0 },   // [1]
    {0 , 0 , 0 , 0 },   // [2]
    {0 , 0 , 0 , 0 }    // [3]
};
```

```
public int getheight(){
    return this.height;
}

public void setheight(int height){
    this.height = height;
}

public int getwidth(){
    return this.width;
}

public void setwidth(int width){
    this.width = width;
}

public String gettitle(){
    return this.title;
}

public void settitle(String title){
    this.title = title;
}
```

```
private int     height  = 600;
private int     width   = 600;
private String  title   = "2048";
```

* Inheritance :

- Penjelasan :

Penurunan sifat dari *class* yang di *extends*. Jadi semua variabel, fungsi yang dimiliki oleh kelas *parent* nya (yang di *extends*).

- Bagian Program yang menggunakan Inheritance :

```
public class window extends JFrame implements KeyListener{
    private int     height  = 600;
    private int     width   = 600;
    private String  title   = "2048";
    Game    board           = new Game();
    JLabel  boardTile[][]   = new JLabel[4][4];
    Border  border          = BorderFactory.createLineBorder(Color.BLUE,5);
```

* Overloading :

- Penjelasan :

Penggunaan suatu fungsi dengan nama yang sama, tetapi memiliki perbedaan parameter.

- Bagian Program yang menggunakan Overloading.

```
// overloading ------------------------------------------------------------
    public int[][] rotateRight(){ //memutar tabel -90 derajat
        int copyTable[][] = new int[4][4];
        for(int row = 0;row<4;row++){
            for(int column=0;column<4;column++){
                copyTable[column][3-row] = boardLabel[row][column];
            }
        }
        return copyTable;
    }

    public int[][] rotateRight(int[][] table){ //memutar tabel -90 derajat
        int copyTable[][] = new int[4][4];
        for(int row = 0;row<4;row++){
            for(int column=0;column<4;column++){
                copyTable[column][3-row] = table[row][column];
            }
        }
        return copyTable;
    }
//  ------------------------------------------------------------
```

*Override :

- Penjelasan :

  Menggunakan fungsi tersebut dari *class* yang di pakai (ditumpuk). Dimana kelas yang di *override* tersebut adalah kelas *abstract*, sehingga harus di tumpuk.

- Bagian Program yang menggunakan Override :

```
@Override
public void keyReleased(KeyEvent e) {

}

@Override
public void keyTyped(KeyEvent e) { //g pakai sih wkwk ,

}

@Override
public void keyPressed(KeyEvent e) {
    int[][] copy = board.copyTableData();
    int keyCode = e.getKeyCode();
    if( keyCode == KeyEvent.VK_UP) {  //jika tombol atas
```

*Event Handling (Adapter/Listener) :

- Penjelasan :

  Jika ada class yang mengimplementasi / menambahkan listener , maka dapat menerima berbagai macam inputan tergantung nama Listener / adapter , mis. KeyListener di gunakan jika ada berbagai inputan dari keyboard berupa , tertekan , terketik , dan terlepas.

- Bagian Program yang menggunakan Override :

```
public class window extends JFrame implements KeyListener{
    private int     height  = 600;
    private int     width   = 600;
    private String  title   = "2048";
    Game    board           = new Game();
    JLabel  boardTile[][]   = new JLabel[4][4];
    Border  border          = BorderFactory.createLineBorder(Color.BLUE,5);
```

```
  @Override
  public void keyReleased(KeyEvent e) {

  }

  @Override
  public void keyTyped(KeyEvent e) { //g pakai sih wkwk ,

  }

  @Override
  public void keyPressed(KeyEvent e) {
      int[][] copy = board.copyTableData();
      int keyCode = e.getKeyCode();
      if( keyCode == KeyEvent.VK_UP) {  //jika tombol atas
```

```
 @Override
 public void actionPerformed(ActionEvent e) { //jika
     board.reset();  // board kita reset jadi 000 sem
     board.start();  // kemudian di ulang kembali mer
     update();       // kita tampilkan di window
     resetFrame.setVisible(false); // frame nya kita
 }
```

## - Fungsi – fungsi yang digunakan di dalam Program “2048” :

### * Class window :

- *Constructor ()*(Untuk menampilkan Window, mensetting bentuk layout di dalam window  dan memasukan JLabel Window) :

```
window(){ //memunculkan window (lebih tepatnya inisialisasi awal)
    setSize(getwidth(),getheight()); //setting ukuran window
    setDefaultCloseOperation(EXIT_ON_CLOSE); //Setting tombol close untuk dapat m
    setResizable(false); // mensetting agar tak dapat di ubah ukuran nya
    setTitle(gettitle()); // set judul window ... variabel judul ada di atas kala
    setLayout(new GridLayout(4,4));// mensetting bentuk agar menjadi bentuk grid
    addKeyListener(this); //menambahkan window agar key keyboard dan window dapat
    for(int row=0;row<4;row++){
        for(int column=0;column<4;column++){
            boardTile[row][column] = new JLabel("",SwingConstants.CENTER); //Memb
            boardTile[row][column].setBorder(border); // men setting agar JLabel
            boardTile[row][column].setFont(new Font("",Font.PLAIN,24)); //mengese
            add(boardTile[row][column]); //barulah di tambahkan JLabel yang sudah
        }
    }
    setVisible(true); //memunculkan grid nya
    board.start(); // starter 2048 yaitu memasukan 2 anggka random 2 / 4 di grid
    update(); // mengupdate tampilan grid di dalam window
}
```

- Get Setter Height,width, dan title (Untuk mendapatkan value / menset value variael yang di *private*) :

```
public int getheight(){
    return this.height;
}

public void setheight(int height){
    this.height = height;
}

public int getwidth(){
    return this.width;
}

public void setwidth(int width){
    this.width = width;
}

public String gettitle(){
    return this.title;
}

public void settitle(String title){
    this.title = title;
}
```

- Update ()(Untuk mengupdate tampilan JLabel dengan menkopi array di *class Game*) :

```
public void update(){ //Pengupdate bentuk grid ke windows
    for(int row=0;row<4;row++){
        for(int column=0;column<4;column++){
            boardTile[row][column].setText(String.valueOf(board.getBoardLabel()[row][column]));
            // boardTile[row][column].setForeground(new Color(100,200,30)); //kemudian mengeset
        }
    }
}
```

- *KeyPressed*((KeyEvent e) Untuk mengecek inputan *key* yang di *inputkan* untuk menentukan gerakan dalam array game, serta mengecek apakah game telah berakhir atau belum):

```
@Override
public void keyPressed(KeyEvent e) {
    int[][] copy = board.copyTableData();
    int keyCode = e.getKeyCode();
    if( keyCode == KeyEvent.VK_UP) {  //jika tombol atas
        board.up3();                  //melakukan fungsi up3
    }
    else if(keyCode == KeyEvent.VK_DOWN){ //jika tombol bawah
        board.down();
    }
    else if(keyCode == KeyEvent.VK_LEFT){ //jika tombol kiri
        board.left();
    }
    else if(keyCode == KeyEvent.VK_RIGHT){ //jika tombol kanan
        board.right();
    }
    if(board.boardSpawnerChecker() == true || board.possibleMove() == true){
        if(Arrays.deepEquals(board.getBoardLabel(),copy)){  // jika tidak ada
            //bruh do nothingggg wkwk
        }else{ //nah jika ada yang bergerak
            board.spawnNumber(); //memunculkan angka 2 / 4 di tempat yang kos
            update();             // update windows untuk menyesuaikan dengan
            // board.printBoard();  //ini cuma untuk print di versi terminal
        }
    }else{
        System.out.println("GAME OVER"); //jika tidak ada tempat dan tidak ad
        GameOver();
    }
}
```

- GameOver() (Disaat game berakhir, maka akan memunculkan window baru untuk me*reset* array game):

```
public void GameOver(){ // saat game berakhir
    JFrame resetFrame = new JFrame("GAME OVER"); //mengin
    JButton resetButton = new JButton("RESET"); // mengin
    resetFrame.setSize(300,100);  // mengeset ukuran fram
    resetButton.addActionListener(new ActionListener(){ /

        @Override
        public void actionPerformed(ActionEvent e) { //ji
            board.reset();  // board kita reset jadi 000
            board.start();  // kemudian di ulang kembali
            update();       // kita tampilkan di window
            resetFrame.setVisible(false); // frame nya ki
        }
    });
    resetFrame.add(resetButton); //kita tambahkan button
    resetFrame.setVisible(true); // menampilkan nya saat
}
```

* **Class Game :**

- boardSpawnerChecker() (Untuk mengecek apakah ada *space* tersisa untuk *spawn* angka):

```
public boolean boardSpawnerChecker(){ //untuk
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            if (boardLabel[row][column] == 0){
                return true;
            }
        }
    }
    return false;
}
```

- spawnNumber() (Mengespawn nomor 2 / 4 di tempat yang kosong) :

```
public void spawnNumber(){ //untuk mengespawn nomor 2 / 4 di tile yang kosong
    int row;
    int column;
    do{
        row     = rand.nextInt(4);
        column  = rand.nextInt(4);
    }while(boardLabel[row][column] != 0); //dilakukan perandoman koordinat untuk pemasukan tile
    int twoOrFour;
    do{
        twoOrFour = rand.nextInt(5);
    }while(twoOrFour!=2 && twoOrFour !=4); //perandom agar nilai yang ingin di tampilkan di koo
    boardLabel[row][column] = twoOrFour;
}
```

- up3() ( Untuk memindahkan semua isi array (boardLabel) ke atas dengan logic permainan 2048 (jika sama di tambah, jika sudah ditambah tidak bisa di tambah, dan jika belum atas dan atasnya kosong angka nya di naikan ):

```
public void up3(){ //versi ke 3 fungsi pada saat user menekan tombol atas ^
    // int[][] copyTable = movedOrNot();
    for (int column = 0;column<4;column++){
        for(int row = 0;row<3;row++){
            for(int ulang = 0;ulang<3;ulang++){ // pertama yang dilakukan adalah memindahkan semu
                for(int fillRow=0+row;fillRow<3;fillRow++){
                    if(boardLabel[fillRow][column] == 0 && boardLabel[fillRow+1][column] !=0){
                        boardLabel[fillRow][column] += boardLabel[fillRow+1][column];
                        boardLabel[fillRow+1][column]=0;
                    }
                }
            }
            if(boardLabel[row][column]==boardLabel[row+1][column] && boardLabel[row][column]!=0
                && boardLabel[row+1][column] != 0)
            {
                boardLabel[row][column]+=boardLabel[row+1][column];
                boardLabel[row+1][column]=0;
            }
        }
    }
}
```

- right() (Sama seperti up3, hanya saja ini untuk memindahkan ke kanan, dengan logic yang sama , kita memutar array (boardLabel) di putar ke kiri , dirubah sesuai dengan fungsi atas, kemudian di putar ke kanan kembali):

```
public void right(){ //fungsi saat user mennekan tombol kanan
    boardLabel = rotateLeft(); //kita putar ke kiri agar bagian kanan menjadi di atas , dan kita
    for (int column = 0;column<4;column++){
        for(int row = 0;row<3;row++){
            for(int ulang = 0;ulang<3;ulang++){
                for(int fillRow=0+row;fillRow<3;fillRow++){
                    if(boardLabel[fillRow][column] == 0 && boardLabel[fillRow+1][column] !=0){
                        boardLabel[fillRow][column] += boardLabel[fillRow+1][column];
                        boardLabel[fillRow+1][column]=0;
                    }
                }
            }
            if(boardLabel[row][column]==boardLabel[row+1][column] && boardLabel[row][column]!=0
            && boardLabel[row+1][column] != 0)
            {
                boardLabel[row][column]+=boardLabel[row+1][column];
                boardLabel[row+1][column]=0;
            }
        }
     }
    boardLabel = rotateRight(); //setelah itu kita kembalikan agar board normal
}
```

- left() (Sama seperti right, hanya saja di putar ke kanan):

```
public void left(){ //Fungsi saat user menekan tombol kiri <--
    boardLabel = rotateRight();
    for (int column = 0;column<4;column++){
        for(int row = 0;row<3;row++){
            for(int ulang = 0;ulang<3;ulang++){
                for(int fillRow=0+row;fillRow<3;fillRow++){
                    if(boardLabel[fillRow][column] == 0 && boardLabel[fillRow+1][column] !=0){
                        boardLabel[fillRow][column] += boardLabel[fillRow+1][column];
                        boardLabel[fillRow+1][column]=0;
                    }
                }
            }
            if(boardLabel[row][column]==boardLabel[row+1][column]
            && boardLabel[row][column]!=0 && boardLabel[row+1][column] != 0)
            {
                boardLabel[row][column]+=boardLabel[row+1][column];
                boardLabel[row+1][column]=0;
            }
        }
     }
    boardLabel = rotateLeft();
}
```

- down() (Sama seperti right, hanya saja di putar balik):

```
public void down(){ //fungsi saat user mennekan tombol bawah
    boardLabel = mirror();
    for (int column = 0;column<4;column++){
        for(int row = 0;row<3;row++){
            for(int ulang = 0;ulang<3;ulang++){
                for(int fillRow=0+row;fillRow<3;fillRow++){
                    if(boardLabel[fillRow][column] == 0 && boardLabel[fillRow+1][column] !=0){
                        boardLabel[fillRow][column] += boardLabel[fillRow+1][column];
                        boardLabel[fillRow+1][column]=0;
                    }
                }
            }
            if(boardLabel[row][column]==boardLabel[row+1][column] && boardLabel[row][column]!=0
            && boardLabel[row+1][column] != 0)
            {
                boardLabel[row][column]+=boardLabel[row+1][column];
                boardLabel[row+1][column]=0;
            }
        }
     }
    boardLabel = mirror();
}
```

- rotateLeft() ( memutar array (boardLabel) 90 derajat ):

```
public int[][] rotateLeft(){ //memutar tabel 90 derajat
    int copyTable[][] = new int[4][4];
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            copyTable[3-column][row] = boardLabel[row][column];
        }
    }
    return copyTable;
}
```

- rotateRight() ( memutar array (boardLabel) -90 derajat ):

```
public int[][] rotateRight(){ //memutar tabel -90 derajat
    int copyTable[][] = new int[4][4];
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            copyTable[column][3-row] = boardLabel[row][column];
        }
    }
    return copyTable;
}
```

- rotateRight(int[][] table) ( memutar table -90 derajat ):

```
public int[][] rotateRight(int[][] table){ //memutar tabel
    int copyTable[][] = new int[4][4];
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            copyTable[column][3-row] = table[row][column];
        }
    }
    return copyTable;
}
```

- mirror() (memutar array (boardLabel) 180 derajat):

```
public int[][] mirror(){ //memutar tabel 180 derajat
    int copyTable[][] = new int[4][4];
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            copyTable[3-row][3-column] = boardLabel[row][column];
        }
    }
    return copyTable;
}
```

- copyTableData() (Mengkopi boardTabel):

```
public int[][] copyTableData(){ //untuk copy boardTable
    int copyTable[][] = new int[4][4];
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            copyTable[row][column] = boardLabel[row][column];
        }
    }
    return copyTable;

}
```

- possibleMove() (Mengecek apakah masih bisa digerakan atau tidak, jika ada akan me return True, jika tidak akan me return False):

```
public boolean possibleMove(){ //mengecek apakah masih bisa di gerakan atau tidak
    int copyTable[][] = copyTableData();
    for (int putar=0;putar<3;putar++){
        for (int column = 0;column<4;column++){
            for(int row = 0;row<3;row++){
                for(int ulang = 0;ulang<3;ulang++){
                    for(int fillRow=0+row;fillRow<3;fillRow++){
                        if(copyTable[fillRow][column] == 0 && copyTable[fillRow+1][column] !=0){
                            copyTable[fillRow][column] += copyTable[fillRow+1][column];
                            copyTable[fillRow+1][column]=0;
                            return true; //Jika bisa di gerakan , langsung return True
                        }
                    }
                }
                if(copyTable[row][column]==copyTable[row+1][column] && copyTable[row][column]!=0
                && copyTable[row+1][column] != 0)
                {
                    copyTable[row][column]+=copyTable[row+1][column];
                    copyTable[row+1][column]=0;
                }
            }
        }
        copyTable = rotateRight(copyTable); //kita putar 3x agar semua kemungkinan gerakan dapat k
    }
    return false; //jika sampai akhir tidak bisa , ya false
}
```

- getBoardLabel() (Untuk mendapatkan nilai dari boardLabel):

```
public int[][] getBoardLabel(){
   return this.boardLabel;
}
```

- start() (Untuk menginisiasi array (boardLabel) dengan merandom 2 angka di koordinat random) :

```
public void start(){ //inisialisasi ke
    for(int ulang=0;ulang<2;ulang++){
        spawnNumber();
    }
}
```

- reset() (Untuk mereset array (boardLabel) menjadi 0 semua):

```
public void reset(){// mereset agar kotak menj
    for(int row = 0;row<4;row++){
        for(int column=0;column<4;column++){
            boardLabel[row][column] = 0;
        }
    }
}
```

- printBoard() (Menampilkan bentuk board di Terminal) :

```
public void printBoard(){ //secara default akan mengambil board
    for(int row = 0;row<4;row++){
        for(int column = 0; column<4;column++){
                System.out.print(getBoardLabel()[row][column]);
            }
            System.out.println();
        }
        System.out.print("---------\n");
}
```

**Referensi :**

https://www.javatpoint.com/abstract-class-in-java

JENI-INTRO