

**Projek UAS**  
**PRAKTIKUM**  
**PEMROGRAMAN BERORIENTASI OBJEK**



**S1 TEKNIK INFORMATIKA**

**Dosen Pengampu :**

Sayekti Harits Suryawan, S.Kom., M.Kom.

**Oleh :**

**Kelompok 8**

Alfi Sukmanata (2211102441168)

M Afif Aunur Rohman (2211102441185)

Debby Fahrizal Rahman (2211102441196)

**UNIVERSITAS MUHAMMADIYAH KALIMANTAN  
TIMUR FAKULTAS SAINS & TEKNOLOGI  
2023/2024**

## Game

Game ini merupakan game simple yang terinspirasi dari game "Breakout" yang luncurkan pada mei 13, 1976. Dalam game ini, player mengontrol sebuah paddle untuk mengarahkan bola dan menghancurkan blok-blok yang ada di atasnya. Bola dapat memantul di dinding dan paddle, dan tujuan utama player adalah menjaga bola agar tidak jatuh ke bagian bawah layar. Player kehilangan nyawa jika bola mencapai bagian bawah layar, dan permainan berakhir jika nyawa habis. Player dapat menggerakkan paddle ke kiri atau kanan menggunakan tombol panah, dan bola dapat dilepaskan dari paddle menggunakan tombol spasi. Terdapat juga elemen tambahan seperti efek asap ketika bola bergerak dan papan skor untuk menghitung skor player.



## World

### Class Board

Class Board merupakan kelas utama yang mengimplementasikan dunia permainan pada permainan Breakout yang dibuat dengan menggunakan Greenfoot. Kelas ini memiliki atribut seperti objek Paddle, variabel untuk jumlah nyawa (heart), objek Counter untuk menghitung dan menampilkan skor, serta objek GreenfootSound untuk memainkan musik latar belakang. Konstruktur kelas Board inisialisasi beberapa objek, seperti Paddle, Counter, dan objek-objek Block sebagai elemen permainan. Metode prepare() digunakan untuk menyiapkan dan menempatkan blok-blok dalam dunia permainan. Kelas ini juga memiliki metode untuk mengatur dan mendapatkan jumlah nyawa, menambah skor, mengatur musik latar belakang, serta menangani peristiwa seperti bola keluar dan permainan berakhir (game over).

The screenshot shows a Java IDE window titled "Board - Projek UAS Batako". The menu bar includes "Class", "Edit", "Tools", and "Options". A toolbar below the menu has buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". On the right side of the interface, there is a "Source Code" dropdown menu.

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Kelompok 8
 * Alfie Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Anur Rohman - 2211102441185
 */
public class Board extends World
{
    private Paddle paddle; //Objek Paddle yang akan digunakan dalam dunia.
    private int heart = 3; //Variabel untuk menyimpan jumlah nyawa atau health awal.
    private Counter scoreCounter; //Objek Counter untuk menghitung dan menampilkan skor.
    GreenfootSound music = new GreenfootSound("song.mp3"); //Objek suara untuk musik latar belakang.

    public Board() //Konstruktor untuk kelas Board.
    {
        super(800, 600, 1);
        started(); //Untuk memulai musik latar belakang.
        scoreCounter = new Counter("Score: "); //Untuk skor dan menambahkannya ke dunia.
        addObject(scoreCounter, 80, 20); //Mengatur posisi scoreCounter
        paddle = new Paddle(); //Menambahkan paddle ke world.
        addObject(paddle, paddle.getWidth() / 2, getHeight() - 40);
        prepare(); //Untuk menyiapkan blok-blok di world.
    }

    public void setHeart(int heart){ //Metode untuk mengatur jumlah nyawa.
        this.heart = heart; //Digunakan untuk mengupdate jumlah nyawa pada dunia.
    }

    public int getHeart(){ //Metode untuk mendapatkan jumlah nyawa.
        return heart; //Digunakan untuk mendapatkan nilai jumlah nyawa pada dunia.
    }

    public void ballOut() //Metode untuk memanggil metode newBall() pada objek Paddle.
    {
        paddle.newBall();
    }

    private void prepare() //Metode untuk menyiapkan blok-blok pada dunia.
    //Membuat dan menambahkan beberapa objek Block ke dunia pada posisi yang ditentukan.
    //Membuat dan menambahkan objek Health (nyawa) ke dunia.
    {
        Block block = new Block();
        addObject(block, 326, 199);
        Block block2 = new Block();
        addObject(block2, 534, 248);
        Block block3 = new Block();
        addObject(block3, 531, 135);
        Block block4 = new Block();
        addObject(block4, 262, 392);
        Block block5 = new Block();
        addObject(block5, 458, 253);
        Block block6 = new Block();
        addObject(block6, 476, 248);
        block6.setLocation(485, 319);
        block.setLocation(398, 83);
        block.setLocation(399, 52);
        block3.setLocation(447, 75);
        block5.setLocation(343, 74);
        block3.setLocation(451, 75);
        block5.setLocation(352, 75);
        block4.setLocation(302, 99);
        block2.setLocation(403, 100);
        block2.setLocation(399, 99);
        block6.setLocation(499, 99);
        block6.setLocation(498, 98);
        block6.setLocation(498, 99);
        Block block7 = new Block();
        addObject(block7, 259, 123);
        Block block8 = new Block();
        addObject(block8, 361, 123);
        block7.setLocation(258, 122);
        block8.setLocation(356, 122);
        Block block9 = new Block();
        addObject(block9, 459, 123);
        block9.setLocation(459, 123);
        block9.setLocation(455, 122);
        Block block10 = new Block();
        addObject(block10, 557, 128);
        block10.setLocation(554, 122);
        Block block11 = new Block();
        addObject(block11, 286, 147);
        Block block12 = new Block();
        addObject(block12, 319, 153);
        Block block13 = new Block();
        addObject(block13, 419, 153);
        block12.setLocation(313, 149);
    }
}
```

```
block19.setLocation(457, 169);
Block block20 = new Block();
addObject(block20, 558, 168);
block20.setLocation(557, 169);
Block block21 = new Block();
addObject(block21, 195, 197);
block21.setLocation(205, 191);
Block block22 = new Block();
addObject(block22, 310, 185);
Block block23 = new Block();
addObject(block23, 310, 185);
block23.setLocation(413, 199);
block22.setLocation(305, 191);
block21.setLocation(206, 191);
block23.setLocation(403, 191);
Block block24 = new Block();
addObject(block24, 506, 196);
block24.setLocation(501, 191);
Block block25 = new Block();
addObject(block25, 656, 178);
block25.setLocation(656, 169);
Block block26 = new Block();
block11.setLocation(206, 146);
block12.setLocation(305, 146);
block13.setLocation(404, 146);
Block block14 = new Block();
addObject(block14, 506, 147);
block14.setLocation(501, 147);
block14.setLocation(504, 145);
block14.setLocation(503, 146);
Block block15 = new Block();
addObject(block15, 604, 148);
block15.setLocation(602, 146);
Block block16 = new Block();
addObject(block16, 168, 177);
block16.setLocation(161, 169);
Block block17 = new Block();
addObject(block17, 265, 169);
block17.setLocation(260, 169);
Block block18 = new Block();
addObject(block18, 363, 169);
block18.setLocation(359, 169);
Block block19 = new Block();
addObject(block19, 462, 168);
addObject(block26, 606, 192);
block26.setLocation(601, 191);
Block block27 = new Block();
addObject(block27, 256, 218);
Block block28 = new Block();
addObject(block28, 357, 217);
block27.setLocation(257, 214);
block28.setLocation(358, 214);
block27.setLocation(259, 214);
Block block29 = new Block();
addObject(block29, 462, 213);
Block block30 = new Block();
addObject(block30, 548, 226);
block29.setLocation(457, 214);
block30.setLocation(555, 214);
Block block31 = new Block();
addObject(block31, 324, 248);
block31.setLocation(303, 235);
Block block32 = new Block();
addObject(block32, 406, 234);
block32.setLocation(403, 235);
Block block33 = new Block();
addObject(block33, 507, 236);
block33.setLocation(507, 236);
block32.setLocation(408, 236);
block31.setLocation(308, 236);
Block block34 = new Block();
addObject(block34, 361, 259);
Block block35 = new Block();
addObject(block35, 455, 266);
block34.setLocation(361, 257);
block35.setLocation(460, 257);
Block block36 = new Block();
addObject(block36, 416, 282);
block36.setLocation(414, 277);
block.setLocation(402, 56);
block36.setLocation(409, 277);
block3.setLocation(453, 76);
block5.setLocation(353, 78);
block3.setLocation(455, 79);
block5.setLocation(355, 79);
block.setLocation(404, 60);
block4.setLocation(302, 101);
block2.setLocation(396, 101);
```

```

        block6.setLocation(494, 101);
        block5.setLocation(352, 81);
        block3.setLocation(455, 82);
        block5.setLocation(354, 82);
        block3.setLocation(451, 82);
        block7.setLocation(258, 127);
        block8.setLocation(356, 126);
        block9.setLocation(455, 126);
        block10.setLocation(556, 127);
        block9.setLocation(456, 126);
        block8.setLocation(358, 127);
        block9.setLocation(456, 127);
        block10.setLocation(555, 127);
        Health health = new Health();
        addObject(health, 716, 20);
        Health health2 = new Health();
        addObject(health2, 740, 20);
        Health health3 = new Health();
        addObject(health3, 764, 20);
    }

    public void gameOver() //Metode untuk menampilkan layar Game Over.
    {
        Greenfoot.delay(5);
        addObject(new ScoreBoard(scoreCounter.getValue()), getWidth()/2, getHeight()/2);
        Greenfoot.playSound("game-over.mp3"); //Memainkan suara Game Over.
        Greenfoot.stop(); //Menghentikan permainan dan menampilkan objek ScoreBoard dengan skor akhir.
    }

    public void countScore(int count) //Metode untuk menambahkan skor.
    {
        scoreCounter.add(count); //Memanggil metode add(count) pada objek scoreCounter untuk menambah skor.
    }

    public void started() //Metode untuk memulai musik latar belakang saat permainan dimulai.
    {
        music.setVolume(35); //mengatur volume musik
        music.playLoop(); //membuat musik mengalami looping
    }

    public void stopped() //Metode untuk menghentikan musik latar belakang saat permainan berakhir.
    {
        music.stop(); //musik berhenti
    }
}

Class compiled - no syntax errors

```



## Actor

### Class Ball

Class Ball adalah representasi dari bola dalam permainan Breakout yang dibuat dengan Greenfoot. Bola memiliki atribut seperti perubahan posisi pada sumbu X dan Y (changeInX dan changeInY), jumlah nyawa (healthCount), status apakah bola terikat pada paddle atau tidak (stuck), dan variabel hitungan untuk efek asap (count). Metode act() mengatur perilaku bola, memeriksa apakah bola tidak terikat dan memanggil metode move(), makeSmoke(), dan checkOut(). Metode move() menggerakkan bola, memeriksa interaksi dengan paddle, dinding, dan blok. Bola dapat memantul dari dinding dan paddle, dan jika menyentuh blok, akan membalikkan arah, memainkan suara, dan menambah skor. Terdapat juga metode untuk memeriksa jika bola berada di luar batas bawah layar, membuat efek asap, memeriksa jika bola bersentuhan dengan paddle atau blok, serta metode untuk melepaskan bola dari paddle.

The screenshot shows the Greenfoot IDE interface with the title bar "Ball - Projek UAS Batako". The menu bar includes "Class", "Edit", "Tools", and "Options". Below the menu is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A "Source Code" dropdown menu is open. The main window displays the Java code for the "Ball" class:

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Kelompok 8
 * Alfie Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Aunur Rohman - 2211102441185
 */
public class Ball extends Actor
{
    private int changeInX; //Variabel yang menyimpan perubahan posisi (kecepatan) bola pada sumbu X.
    private int changeInY; //Variabel yang menyimpan perubahan posisi (kecepatan) bola pada sumbu Y.
    private int healthCount; //Variabel yang menyimpan jumlah nyawa atau health.
    private boolean stuck = true; //Variabel yang menunjukkan apakah bola stuck pada paddle atau tidak.
    private int count = 2; //Variabel penghitung untuk membuat asap (smoke) setiap beberapa iterasi.
    private GreenfootSound sound = new GreenfootSound("proton.mp3"); //Objek suara untuk efek suara saat bola memantul.

    /**
     * Act. Bergerak jika tidak mengalami stuck.
     */
    public void act() //Metode yang dipanggil setiap kali tombol "Act" atau "Run" ditekan dalam lingkungan permainan.
    {
        if (!stuck)
```

```

        {
            move();
            makeSmoke();
            checkOut();
            //Jika bola tidak terikat pada paddle, maka panggil metode move(), makeSmoke(), dan checkOut().
        }
    }

    /**
     * Pindahkan bola. Kemudian memeriksa apakah bola terkena hit.
     */
    public void move() //Metode untuk memindahkan posisi bola dan memeriksa interaksi dengan objek lain.
    {
        /** setLocation memindahkan posisi bola, dengan menempatkannya
         * pada posisi lamanya.
         */
        setLocation (getX() + changeInX, getY() + changeInY); //Memindahkan posisi bola berdasarkan perubahan kecepatan pada sumbu X dan Y.
        checkPaddle(); //Memeriksa interaksi dengan paddle.
        checkWalls(); //Memeriksa jika bola bertabrakan dengan dinding.
        checkBlock(); //Memeriksa interaksi dengan blok.
    }

    /**
     * Memeriksa apakah bola menyentuh paddle, lalu menyesuaikan pergerakannya.
     */
    private void checkPaddle() //Metode untuk memeriksa apakah bola bersentuhan dengan paddle.
    {
        Actor paddle = getOneIntersectingObject(Paddle.class); //Mendapatkan objek paddle yang bersentuhan dengan bola.
        if (paddle != null) {
            //Ika terdapat paddle:
            //Membalikkan pergerakan bola pada sumbu Y (changeInY = -changeInY).
            //Menghitung offset dan mempengaruhi pergerakan pada sumbu X berdasarkan posisi bola terhadap paddle.
            changeInY = -changeInY;
            int offset = getX() - paddle.getX();
            changeInX = changeInX + (offset/7);
            if (changeInX > 9) {
                changeInX = 9;
            }
            if (changeInX < -9) {
                changeInX = -9;
            }
        }
    }

    /**
     * Memeriksa apakah kita telah menabrak salah satu dari tiga dinding. Membalikkan arah.
     */
    private void checkWalls() //Metode untuk memeriksa jika bola bersentuhan dengan dinding dunia permanen.
    {
        if (getX() == 0 || getX() == getWorld().getWidth()-1) {
            changeInX = -changeInX; //Jika bola mencapai batas kiri atau kanan dunia, maka membalikkan pergerakan pada sumbu X (changeInX = -changeInX).
        }
        if (getY() == 0) {
            changeInY = -changeInY; //Jika bola mencapai batas atas dunia, maka membalikkan pergerakan pada sumbu Y (changeInY = -changeInY).
        }
    }

    /**
     * Memeriksa apakah kita diluar (Layar bagian bawah)
     */
    private void checkOut() //Metode untuk memeriksa jika bola berada di bagian bawah dunia permanen.
    {
        if (getY() == getWorld().getHeight()-1) { //Ika bola mencapai bagian bawah dunia:
            Greenfoot.playSound("lost_life.mp3"); //Memainkan suara kehilangan nyawa ("lost_life.mp3").
            ((Board) getWorld()).ballOut(); //Memanggil metode ballOut() dari kelas Board untuk mengurangi nyawa dan mengupdate skor.
            healthCount = ((Board) getWorld()).getHeart();
            this.getWorld().removeObjects(getWorld().getObjects(Health.class));
            //Ika jumlah nyawa masih tersisa, menambahkan kembali hati di layar.
            if(healthCount == 3){
                Health health2 = new Health();
                getWorld().addObject(health2,740, 20);
                Health health3 = new Health();
                getWorld().addObject(health3,764,20);
                healthCount -= 1;
                ((Board) getWorld()).setHeart(healthCount);
                getWorld().removeObject(this);
            }
            else if (healthCount == 2){
                Health health3 = new Health();
                getWorld().addObject(health3,764,20);
                healthCount -= 1;
                ((Board) getWorld()).setHeart(healthCount);
                getWorld().removeObject(this);
            }
            else if (healthCount == 1){
                this.getWorld().removeObjects(getWorld().getObjects(Block.class));
            }
        }
    }
}

```

```

        ((Board) getWorld()).gameOver();
        //Jika jumlah nyawa habis, memanggil metode gameOver() dari kelas Board.

    }

}

private void makeSmoke() //Metode untuk membuat efek asap (smoke) setiap beberapa iterasi.
{
    count--; //Mengurangi hitungan setiap iterasi.
    if (count == 0) {
        getWorld().addObject ( new Smoke(), getX(), getY());
        count = 2; //Jika hitungan mencapai 0, maka membuat objek Smoke dan mengatur ulang hitungan.
    }
}

/**
 * Memeriksa jika bola menabrak salah satu blocks. Balikkan arah jika perlu.
 */
private void checkBlock() //Metode untuk memeriksa jika bola bersentuhan dengan blok.
{
    if( !(getWorld().getObjects(Block.class).size() == 0 )) {
        Actor block = getOneIntersectingObject(Block.class);
        //Mendapatkan objek blok yang bersentuhan dengan bola.
        if (block != null) { //Jika terdapat blok:
            changeInY = -changeInY; //Membalikkan pergerakan bola pada sumbu Y (changeInY = -changeInY).
            int offset = getX() - block.getX(); //Menghitung offset dan mempengaruhi pergerakan pada sumbu X berdasarkan posisi
            changeInX = changeInX + (offset/7);
            if (changeInX > 9) {
                changeInX = 9;
            }
            if (changeInX < -9) {
                changeInX = -9;
            }
            sound.setVolume(100);
            sound.play(); //Memainkan suara ("proton.mp3").
            ((Board) getWorld()).countScore(5); //Memanggil metode countScore(5) dari kelas Board untuk menambah skor.
        }
        World world;
        world = getWorld();
        world.removeObject(block); //Menghapus blok dari dunia permainan.
    }
    else {
        ((Board) getWorld()).gameOver();
    }
}

/**
 * Menggerakkan bola ke samping dengan jarak tertentu.
 */
public void move(int dist) //Metode untuk memindahkan posisi bola sejauh dist pada sumbu X.
{
    setLocation (getX() + dist, getY());
}

/**
 * Meluncurkan bola dari paddle.
 */
public void release() //Metode untuk melepaskan bola dari paddle.
{
    changeInX = Greenfoot.getRandomNumber(11) - 5; //Menetapkan perubahan kecepatan bola pada sumbu X dan Y secara acak.
    changeInY = -9;
    stuck = false; //Menetapkan stuck menjadi false sehingga bola tidak lagi terikat pada paddle.
}
}

Class compiled - no syntax errors

```

## Class Block

Kelas Block merupakan representasi dari blok dalam permainan Breakout yang dibuat dengan Greenfoot. Blok ini memiliki atribut kecepatan pergerakan (speed), diinisialisasi dengan nilai 2. Metode act() dijalankan setiap frame permainan dan mengatur pergerakan blok secara horizontal. Blok akan berpindah ke arah yang berlawanan jika sudah mencapai batas kanan atau kiri dunia permainan. Dengan demikian, kelas ini menyediakan logika sederhana untuk pergerakan horizontal blok dalam permainan.

The screenshot shows the Greenfoot IDE interface with the title 'Block - Projek UAS Batako'. The menu bar includes 'Class', 'Edit', 'Tools', and 'Options'. The toolbar has buttons for 'Compile', 'Undo', 'Cut', 'Copy', 'Paste', 'Find...', and 'Close'. A dropdown menu 'Source Code' is open. The code editor contains the following Java code:

```

import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Kelompok 8
 * Alfi Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Aunur Rohman - 2211102441185
 */
public class Block extends Actor
{
    private int speed = 2; //Variabel yang menyimpan nilai kecepatan pergerakan objek Block. Nilainya diatur awalnya ke 2.
    public void act() //Metode yang dipanggil setiap kali tombol "Act" atau "Run" ditekan dalam lingkungan permainan.
    {
        int length = getWorld().getWidth() + 9; //Variabel length diinisialisasi dengan lebar dunia permainan ditambah 9 piksel.
        if (speed > 0 && getX() == getWorld().getWidth() - 1 ||
            (speed < 0 && getX() == 0)) //Pengecekan apakah objek Block berada di batas kanan atau kiri dunia permainan.
            speed = -speed; //Mengurangi kecepatan jika kondisi terpenuhi
        setLocation(getX() + speed, getY()); //Mengubah posisi objek Block berdasarkan kecepatan yang telah dihitung sebelumnya.
    }
}

```

The status bar at the bottom left says 'Class compiled - no syntax errors' and 'saved'. The system tray at the bottom right shows various icons and the date '22/12/2023'.

## Class Counter

Class Counter pada permainan Greenfoot ini digunakan untuk menampilkan nilai numerik di layar, seperti skor atau counter tertentu. Dalam kodennya, terdapat variabel yang menyimpan gambar latar belakang, nilai aktual counter (value), nilai target counter (target) untuk animasi, dan teks prefix yang ditampilkan sebelum nilai counter. Metode act() digunakan untuk menganimasikan tampilan counter menuju nilai target. Metode add(int score) menambahkan skor baru ke nilai target dengan animasi. Metode setValue(int newValue) digunakan untuk mengatur nilai target dan aktual tanpa animasi. Terdapat pula metode untuk mendapatkan nilai target (getValue()) dan mengatur teks prefix (setPrefix(String prefix)). Metode updateImage() memperbarui gambar pada layar sehingga menampilkan nilai terbaru dengan teks prefix yang sesuai. Keseluruhannya, kelas ini memberikan fungsionalitas untuk menampilkan dan mengelola nilai numerik dengan animasi pada layar permainan.

The screenshot shows the Greenfoot IDE interface with the title 'Counter - Projek UAS Batako'. The menu bar includes 'Class', 'Edit', 'Tools', and 'Options'. The toolbar has buttons for 'Compile', 'Undo', 'Cut', 'Copy', 'Paste', 'Find...', and 'Close'. A dropdown menu 'Source Code' is open. The code editor contains the following Java code:

```

import greenfoot.*;

/**
 * Kelompok 8
 * Alfi Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Aunur Rohman - 2211102441185
 *
 * Counter class mengijinkan kita untuk menampilkan nilai numerical di layar.
 */
public class Counter extends Actor
{
    private static final Color transparent = new Color(0,0,0); //Variabel konstan yang menyimpan warna transparan.
    private GreenfootImage background; //Menyimpan gambar latar belakang.
    private int value; //Menyimpan nilai aktual dari counter.
    private int target; //Menyimpan nilai target untuk animasi.
    private String prefix; //Menyimpan teks prefix yang akan ditampilkan sebelum nilai counter.

    public Counter() //Konstruktor tanpa parameter yang memanggil konstruktor lainnya dengan parameter string kosong.
    {
        this(new String());
    }

    if (value < target) {
        value++;
        updateImage(); //Jika nilai aktual kurang dari nilai target, increment nilai aktual dan panggil updateImage().
    }
    else if (value > target) {
    }
}

```

```
        value--;
        updateImage(); //Jika nilai aktual lebih dari nilai target, decrement nilai aktual dan panggil updateImage().
    }

    /**
     * Menambahkan score baru ke nilai counter saat ini. ini akan menganimasikan
     * counter pada frame yang berurutan hingga mencapai nilai baru.
     */
    public void add(int score) //Metode untuk menambahkan nilai ke nilai target.
    {
        target += score; //Menambahkan nilai parameter (score) ke nilai target.
    }

    /**
     * Mengembalikan nilai counter saat ini.
     */
    public int getValue() //Metode untuk mendapatkan nilai target saat ini.
    {
        return target; //Mengembalikan nilai target.
    }

    /**
     * Mengatur nilai counter baru. ini tidak akan menganimasikan counter.
     */
    public void setValue(int newValue) //Metode untuk mengatur nilai target dan nilai aktual.
    {
        target = newValue; //Mengatur nilai target.
        value = newValue; //Mengatur nilai aktual.
        updateImage(); //Memanggil metode updateImage() untuk memperbarui tampilan gambar.
    }

    /**
     * Mengatur teks prefix yang seharusnya ditampilkan sebelumnya
     * nilai counter (e.g. "Score: ").
     */
    public void setPrefix(String prefix) //Metode untuk mengatur teks prefix.
    {
        this.prefix = prefix; //Mengatur teks prefix.
        updateImage(); //Memanggil metode updateImage() untuk memperbarui tampilan gambar.
    }

    /**
     * Memperbarui gambar di layar untuk menampilkan nilai terbaru
     */
    private void updateImage() //Metode untuk memperbarui gambar pada layar sehingga menampilkan nilai terbaru.
    {
        GreenfootImage image = new GreenfootImage(background); //Membuat salinan gambar latar belakang.
        GreenfootImage text = new GreenfootImage(prefix + value, 22, Color.WHITE, transparent); //Membuat gambar teks dengan nilai

        if (text.getWidth() > image.getWidth() - 20)
        {
            image.scale(text.getWidth() + 20, image.getHeight()); //Jika lebar teks lebih besar dari lebar gambar dikurangi 20.
        }

        image.drawImage(text, (image.getWidth()-text.getWidth())/2,
                        (image.getHeight()-text.getHeight())/2); //Menggambar teks di tengah gambar.
        setImage(image); //Mengatur gambar objek Counter dengan gambar yang baru dibuat.
    }
}

Class compiled - no syntax errors
saved

```

## Class Health

Class Health dalam permainan Greenfoot ini tampaknya tidak memiliki implementasi atau fungsi yang jelas dalam kode yang diberikan. Konstruktor kelas ini tidak melakukan operasi khusus, dan metode remove() yang disertakan dalam komentar tidak diimplementasikan dengan jelas. Dengan begitu, pada tahap ini, tidak dapat diidentifikasi fungsi atau tujuan khusus dari kelas Health ini. Jika ada informasi lebih lanjut atau implementasi tambahan yang belum diberikan, mungkin akan lebih mudah memberikan penjelasan yang lebih detail.

Health - Projek UAS Batako

Class Edit Tools Options

Board X Ball X Block X Counter X Health X

Compile Undo Cut Copy Paste Find... Close Source Code

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Kelompok 8
 * Alfi Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Aunur Rohman - 2211102441185
 */
public class Health extends Actor
{
    public Health() //Konstruktor kelas Health.
    {
        //Konstruktor ini tidak memiliki implementasi yang jelas dalam kode yang diberikan.
    }

    public void remove() //Metode yang mencoba untuk menghapus objek dari dunia.
    {
        //getWorld().removeObject().get(0);
    }
}
```

Class compiled - no syntax errors saved

Cari 22/12/2023 16:55

## Class Paddle

Class Paddle merupakan representasi dari objek paddle dalam permainan Greenfoot. Saat paddle ditambahkan ke dalam dunia permainan, sebuah objek bola akan muncul di atasnya menggunakan metode newBall(). Paddle dapat digerakkan ke kiri dan kanan menggunakan tombol panah, dan jika pemain memiliki bola (diidentifikasi melalui metode haveBall()), pemain dapat melepaskan bola dengan menekan tombol spasi. Kelas ini juga menyediakan metode untuk memindahkan paddle, membuat bola baru, dan melepaskan bola yang terkait dengan paddle.

Paddle - Projek UAS Batako

Class Edit Tools Options

Board X Ball X Block X Counter X Health X Paddle X

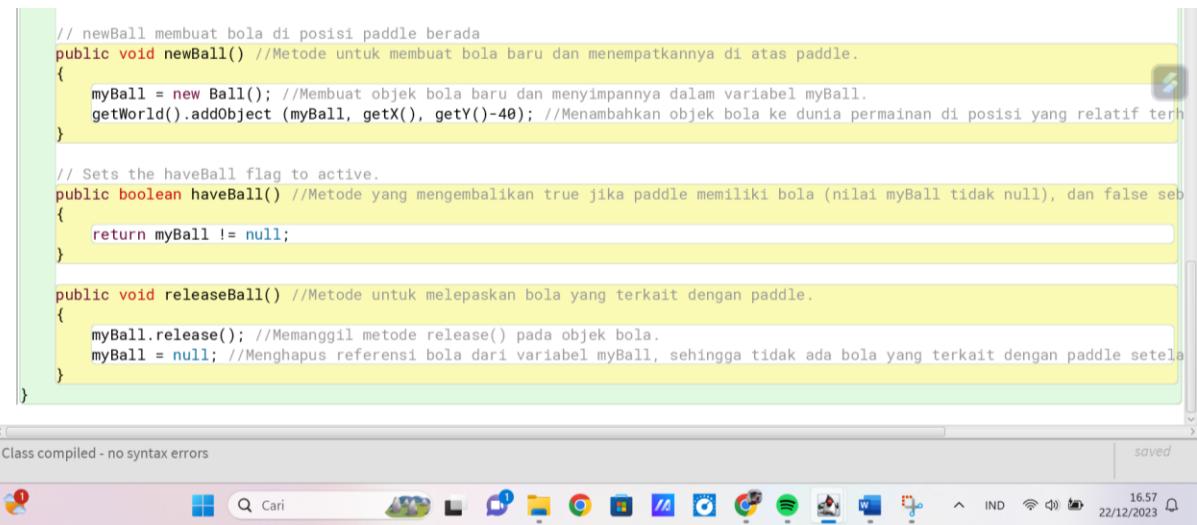
Compile Undo Cut Copy Paste Find... Close Source Code

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Kelompok 8
 * Alfi Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Aunur Rohman - 2211102441185
 */
public class Paddle extends Actor
{
    private Ball myBall; // digunakan sebelum bola di luncurkan
    /**
     * Saat paddle diciptakan, bola juga akan muncul.
     */
    public void addedToWorld(World world) //Metode ini dipanggil saat objek Paddle ditambahkan ke dalam dunia permainan.
    {
        newBall(); //Memanggil metode newBall() untuk membuat dan menempatkan bola di atas paddle.
    }

    /**
     *
     */
    public void act() //Metode ini dipanggil setiap saat Greenfoot melakukan iterasi (frame) dalam permainan.
    {
        if (haveBall() && Greenfoot.isKeyDown ("space")) {
            releaseBall(); //Jika paddle memiliki bola (haveBall() == true) dan tombol spasi ("space") ditekan, maka bola akan di
        }
    }

    public void move(int dist) //Metode untuk memindahkan paddle sejauh dist satuan.
    {
        setLocation (getX() + dist, getY()); //Memindahkan lokasi paddle sejauh dist piksel secara horizontal.
        if (myBall != null) {
            myBall.move (dist); //bola yang terkait dengan paddle) tidak null, maka panggil myBall.move(dist) untuk memindahkan b
        }
    }
}
```



```

// newBall membuat bola di posisi paddle berada
public void newBall() //Metode untuk membuat bola baru dan menempatkannya di atas paddle.
{
    myBall = new Ball(); //Membuat objek bola baru dan menyimpannya dalam variabel myBall.
    getWorld(). addObject (myBall, getX(), getY()-40); //Menambahkan objek bola ke dunia permainan di posisi yang relatif terhadap posisi paddle.
}

// Sets the haveBall flag to active.
public boolean haveBall() //Metode yang mengembalikan true jika paddle memiliki bola (nilai myBall tidak null), dan false sebaliknya.
{
    return myBall != null;
}

public void releaseBall() //Metode untuk melepaskan bola yang terkait dengan paddle.
{
    myBall.release(); //Memanggil metode release() pada objek bola.
    myBall = null; //Menghapus referensi bola dari variabel myBall, sehingga tidak ada bola yang terkait dengan paddle setelah ini.
}

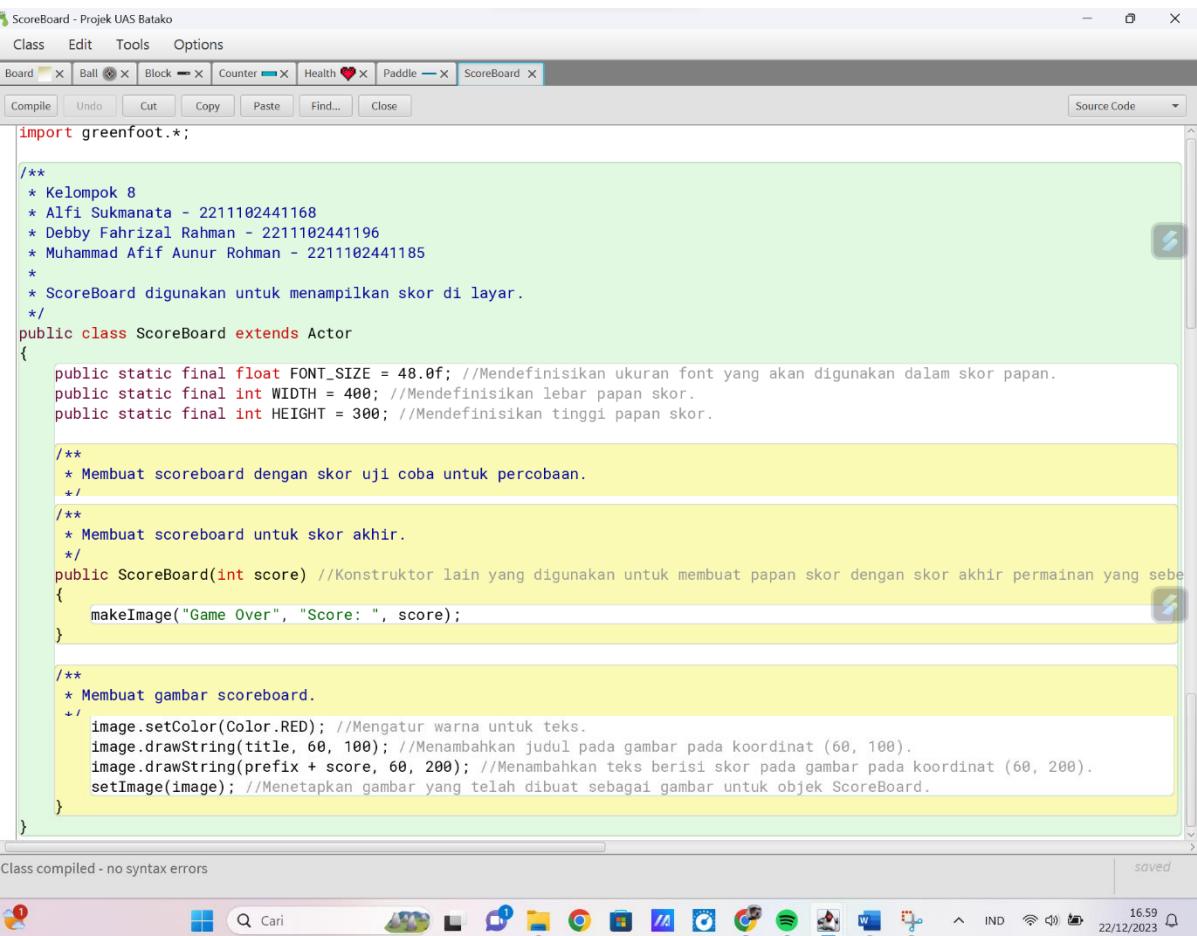
```

Class compiled - no syntax errors

16.57 22/12/2023

## Class ScoreBoard

Class ScoreBoard digunakan untuk menampilkan papan skor di layar permainan Greenfoot. Konstruktor default membuat papan skor dengan skor uji coba, sementara konstruktor lainnya digunakan untuk membuat papan skor dengan skor akhir permainan yang sebenarnya. Metode makeImage bertanggung jawab untuk membuat gambar papan skor dengan judul, teks, dan skor yang diberikan, menggunakan objek GreenfootImage. Papan skor memiliki warna latar belakang biru transparan dengan efek border, serta teks judul dan skor yang ditampilkan dengan warna dan ukuran font yang sesuai.



```

ScoreBoard - Projek UAS Batako
Class Edit Tools Options
Board X Ball X Block X Counter X Health X Paddle X ScoreBoard X
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*;

/**
 * Kelompok 8
 * Alfi Sukmanata - 2211102441168
 * Debby Fahrizal Rahman - 2211102441196
 * Muhammad Afif Aunur Rohman - 2211102441185
 *
 * ScoreBoard digunakan untuk menampilkan skor di layar.
 */
public class ScoreBoard extends Actor
{
    public static final float FONT_SIZE = 48.0f; //Mendefinisikan ukuran font yang akan digunakan dalam skor papan.
    public static final int WIDTH = 400; //Mendefinisikan lebar papan skor.
    public static final int HEIGHT = 300; //Mendefinisikan tinggi papan skor.

    /**
     * Membuat scoreboard dengan skor uji coba untuk percobaan.
     */
    /**
     * Membuat scoreboard untuk skor akhir.
     */
    public ScoreBoard(int score) //Konstruktor lain yang digunakan untuk membuat papan skor dengan skor akhir permainan yang sebenarnya.
    {
        makeImage("Game Over", "Score: ", score);
    }

    /**
     * Membuat gambar scoreboard.
     */
    {
        image.setColor(Color.RED); //Mengatur warna untuk teks.
        image.drawString(title, 60, 100); //Menambahkan judul pada gambar pada koordinat (60, 100).
        image.drawString(prefix + score, 60, 200); //Menambahkan teks berisi skor pada gambar pada koordinat (60, 200).
        setImage(image); //Menetapkan gambar yang telah dibuat sebagai gambar untuk objek ScoreBoard.
    }
}

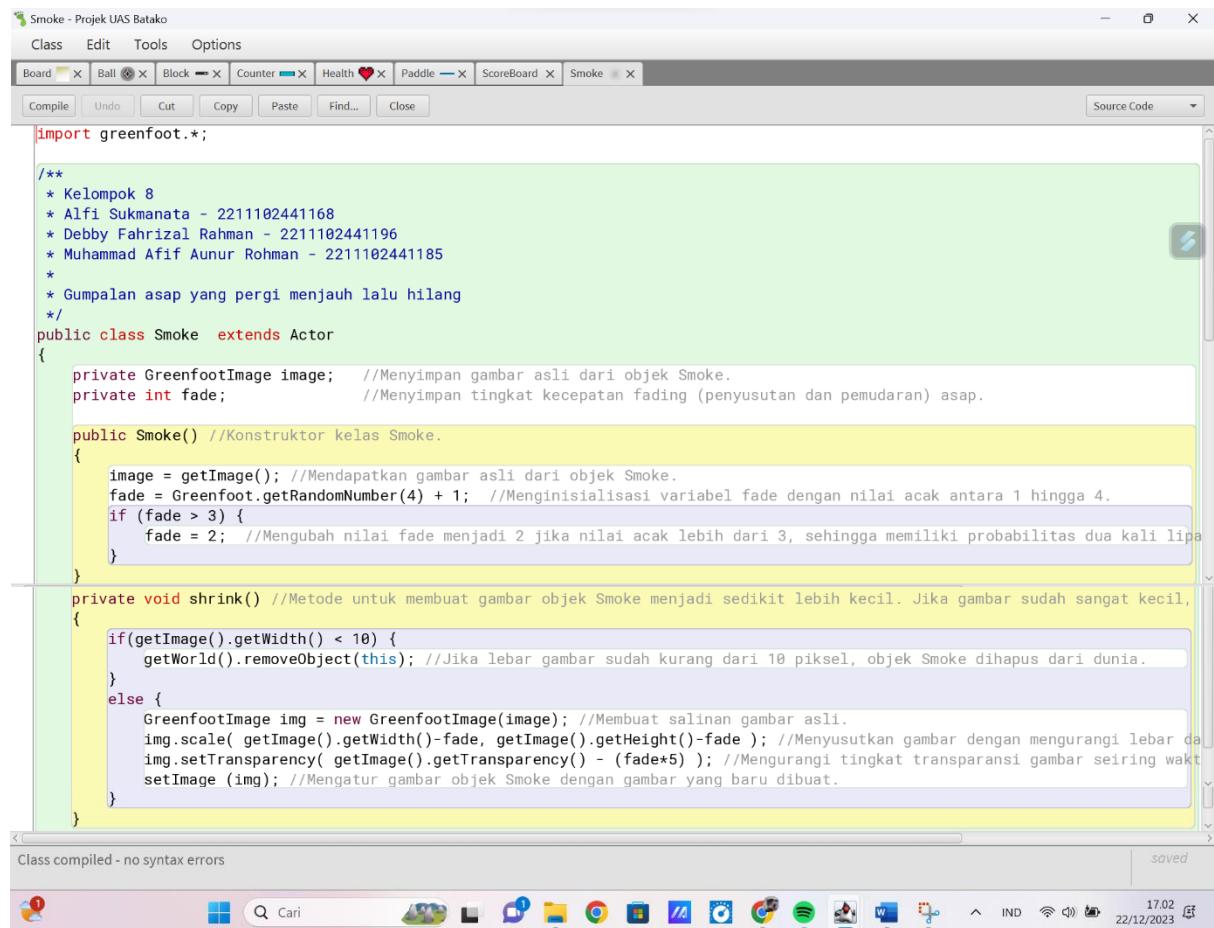
```

Class compiled - no syntax errors

16.59 22/12/2023

## Class Smoke

Class Smoke merupakan representasi dari gumpalan asap dalam permainan Greenfoot. Konstruktor kelas ini menginisialisasi gambar asap dan tingkat kecepatan fading (penyusutan dan pemudaran) asap. Setiap iterasi dalam metode act, gumpalan asap menyusut dengan mengurangi lebar dan tingginya berdasarkan nilai fade. Jika lebar gambar sudah kurang dari 10 piksel, objek Smoke dihapus dari dunia. Proses penyusutan ini juga disertai dengan pengurangan tingkat transparansi, menciptakan efek visual asap yang semakin memudar seiring waktu.



The screenshot shows the Greenfoot IDE interface with the 'Smoke' class selected. The code editor displays the following Java code:

```
import greenfoot.*;  
  
/**  
 * Kelompok 8  
 * Alfi Sukmanata - 2211102441168  
 * Debby Fahrizal Rahman - 2211102441196  
 * Muhammad Afif Aunur Rohman - 2211102441185  
 *  
 * Gumpalan asap yang pergi menjauh lalu hilang  
 */  
public class Smoke extends Actor  
{  
    private GreenfootImage image; //Menyimpan gambar asli dari objek Smoke.  
    private int fade; //Menyimpan tingkat kecepatan fading (penyusutan dan pemudaran) asap.  
  
    public Smoke() //Konstruktur kelas Smoke.  
    {  
        image = getImage(); //Mendapatkan gambar asli dari objek Smoke.  
        fade = Greenfoot.getRandomNumber(4) + 1; //Menginisialisasi variabel fade dengan nilai acak antara 1 hingga 4.  
        if (fade > 3) {  
            fade = 2; //Mengubah nilai fade menjadi 2 jika nilai acak lebih dari 3, sehingga memiliki probabilitas dua kali lipat  
        }  
    }  
  
    private void shrink() //Metode untuk membuat gambar objek Smoke menjadi sedikit lebih kecil. Jika gambar sudah sangat kecil,  
    {  
        if(getImage().getWidth() < 10) {  
            getWorld().removeObject(this); //Jika lebar gambar sudah kurang dari 10 piksel, objek Smoke dihapus dari dunia.  
        }  
        else {  
            GreenfootImage img = new GreenfootImage(image); //Membuat salinan gambar asli.  
            img.scale( getImage().getWidth()-fade, getImage().getHeight()-fade ); //Menyusutkan gambar dengan mengurangi lebar dan  
            img.setTransparency( getImage().getTransparency() - (fade*5) ); //Mengurangi tingkat transparansi gambar seiring waktu  
            setImage (img); //Mengatur gambar objek Smoke dengan gambar yang baru dibuat.  
        }  
    }  
}
```

The status bar at the bottom indicates "Class compiled - no syntax errors" and "saved". The taskbar at the bottom shows various application icons.