

UAS
PEMROGRAMAN BERORIENTASI BERBASIS OBJEK



ANGGOTA KELOMPOK:

- MUHAMMAD RIKY PRADANA_22111024412138
- AHMAD YUSUF MUBARAK_2211102441243
- SUJAINIL AMA BAKKA_2211102441215
- FA'IZDAFFA NAWWARIZZQI MAULANA_2211102441119

S1 Teknik Informatika

Fakultas Saints Dan Teknologi
Universitas Muhammadiyah Kalimantan Timur Prodi Teknik Informatika

Tampilan Game:



```
MyWorld x
Compile Undo Cut Copy Paste Find... Close
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class MyWorld extends World
{
    public MyWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 600, 1);
        prepare();
    }

    private void prepare()
    {
        ambulance ambulance = new ambulance();
        addObject(ambulance, 307, 557);
    }

    public void act(){
        if(Greenfoot.getRandomNumber(100)<1){
            addObject(new carblue(), Greenfoot.getRandomNumber(200) + 200, 0);
        }
        if(Greenfoot.getRandomNumber(500)<1){
            addObject(new carred(), Greenfoot.getRandomNumber(200) + 200, 600);
        }
        if(Greenfoot.getRandomNumber(10)<2){
            addObject(new tree(), Greenfoot.getRandomNumber(170), 0);
            addObject(new tree(), Greenfoot.getRandomNumber(170) + 440, 0);
        }
        if(Greenfoot.getRandomNumber(300)<1){
            addObject(new coin(), Greenfoot.getRandomNumber(200) + 200, 0);
        }
    }
}
```

1. **Impor Greenfoot:**

- Mengimpor paket Greenfoot untuk menggunakan kelas dan metode yang disediakan.

2. **Kelas MyWorld:**

- Mendefinisikan kelas MyWorld yang merupakan turunan dari kelas World dalam Greenfoot.
- Konstruktor kelas memanggil konstruktor kelas induk World dengan parameter lebar, tinggi, dan ukuran sel.

3. **Konstruktor MyWorld:**

- Membuat dunia dengan ukuran 600x600 cells dan ukuran sel 1x1 pixel.
- Memanggil metode prepare() untuk menyiapkan elemen-elemen awal dalam dunia.

4. **Metode Prepare:**

- Metode dipanggil dari konstruktor untuk menyiapkan elemen awal dalam dunia.
- Menempatkan objek ambulance di koordinat (307, 557).

5. **Metode Act:**

- Metode utama yang dipanggil secara berkala selama simulasi berjalan.
- Berisi kondisi untuk menambahkan objek ke dunia secara acak dengan peluang tertentu.
- Dengan peluang 1%, menambahkan objek carblue di posisi acak pada bagian atas dunia.
- Dengan peluang 0.2%, menambahkan objek carred di posisi acak pada bagian bawah dunia.
- Dengan peluang 20%, menambahkan dua objek tree di posisi acak di bagian atas dunia.
 - Dengan peluang 0.33%, menambahkan objek coin di posisi acak pada bagian atas dunia.

MyWorld X ambulance X

CompileUndoCutCopyPasteFind...Close

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

public class ambulance extends Actor
{
    int score = 0;

    public void act()
    {
        getWorld().showText("Score : " + score, 70, 30);
        checkKey();

        end();
        addscore();
    }

    public void checkKey(){
        if(Greenfoot.isKeyDown("left")){
            if(getX() >= 215){
                setLocation(getX() -2, getY());
            }
        }
        if(Greenfoot.isKeyDown("right")){
            if(getX() <= 398){
                setLocation(getX() +2, getY());
            }
        }
        if(Greenfoot.isKeyDown("up")){
            setLocation(getX(), getY() -2);
        }
        if(Greenfoot.isKeyDown("down")){
            setLocation(getX(), getY() +2);
        }
    }

    public void end(){
        if(isTouching(carblue.class) || isTouching(carred.class)){
            getWorld().showText("Game Over \n Score : " + score, 300, 300);
            Greenfoot.stop();
        }
    }

    public void addscore(){
        if(isTouching(coin.class)){
            score = score + 20;
            removeTouching(coin.class);
        }
    }
}
```

1. Variabel Score:

- Mendeklarasikan variabel score dengan tipe data integer dan nilai awal 0.
- Digunakan untuk melacak skor pemain.

2. Metode Act:

- Metode yang dijalankan berkala selama simulasi.
- Menampilkan skor di posisi (70, 30).
- Memanggil metode checkKey() untuk menanggapi input kunci.
- Memanggil metode end() untuk memeriksa sentuhan ambulans dengan objek carblue atau carred yang mengakhiri permainan.
- Memanggil metode addscore() untuk menanggapi sentuhan ambulans dengan objek coin dan menambah skor.

3. Metode CheckKey:

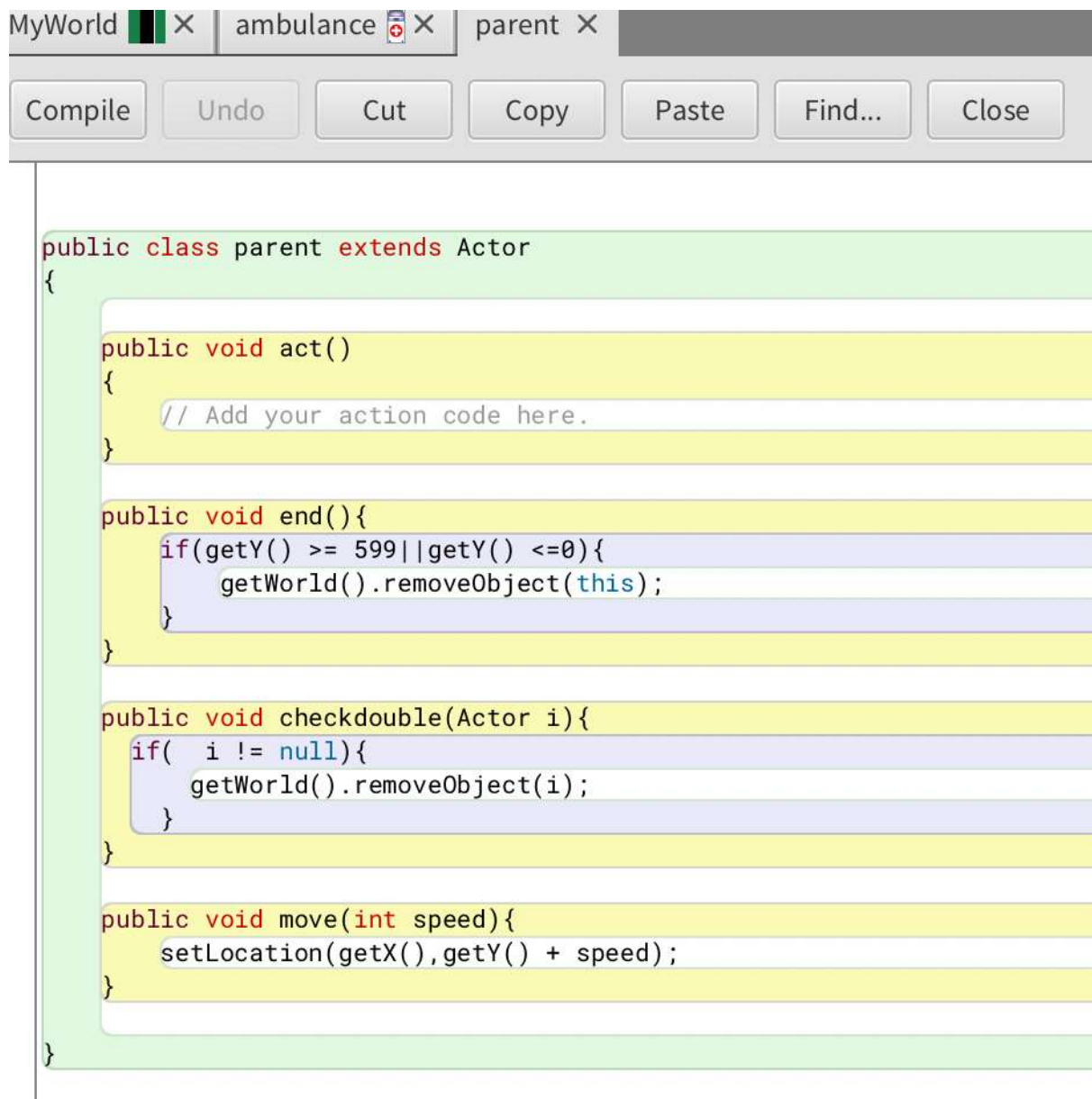
- Menanggapi input kunci dari pemain.
- Menggeser posisi ambulans ke kiri jika tombol panah kiri ditekan dan ambulans belum mencapai batas kiri tertentu.
- Sama halnya untuk tombol panah kanan, atas, dan bawah.

4. Metode End:

- Mengakhiri permainan jika ambulans bersentuhan dengan objek carblue atau carred.
- Menampilkan "Game Over" dan skor saat permainan berakhir.
- Memanggil Greenfoot.stop() untuk menghentikan permainan.

5. Metode Addscore:

- Menambah skor jika ambulans bersentuhan dengan objek coin.
- Menambah 20 poin ke skor dan menghapus objek coin dari dunia.

The image shows a screenshot of a Java IDE window. The title bar at the top contains three tabs: 'MyWorld' with a green icon, 'ambulance' with a red cross icon, and 'parent' with a grey 'X' icon. Below the tabs is a toolbar with buttons for 'Compile', 'Undo', 'Cut', 'Copy', 'Paste', 'Find...', and 'Close'. The main editor area displays the source code for the 'parent' class, which extends the 'Actor' class. The code is color-coded: keywords like 'public', 'class', 'void', 'if', and 'return' are in red; identifiers like 'parent', 'Actor', 'i', and 'this' are in blue; and literals like 'null' are in purple. The code defines four methods: 'act()' with a comment, 'end()' with a boundary check, 'checkdouble()' with a null check, and 'move()' with a location update. The code is enclosed in curly braces to define the class and method scopes.

```
public class parent extends Actor
{
    public void act()
    {
        // Add your action code here.
    }

    public void end(){
        if(getY() >= 599 || getY() <=0){
            getWorld().removeObject(this);
        }
    }

    public void checkdouble(Actor i){
        if( i != null){
            getWorld().removeObject(i);
        }
    }

    public void move(int speed){
        setLocation(getX(),getY() + speed);
    }
}
```

1. Metode Act:

- Metode act yang berjalan berkala selama simulasi.
- Saat ini kosong, pengguna dapat menambahkan kode aksi sesuai kebutuhan.

2. Metode End:

- Memeriksa apakah objek parent berada di luar batas atas atau bawah dunia.
- Jika posisi objek melebihi batas tersebut, objek dihapus dari dunia menggunakan `getWorld().removeObject(this)`.

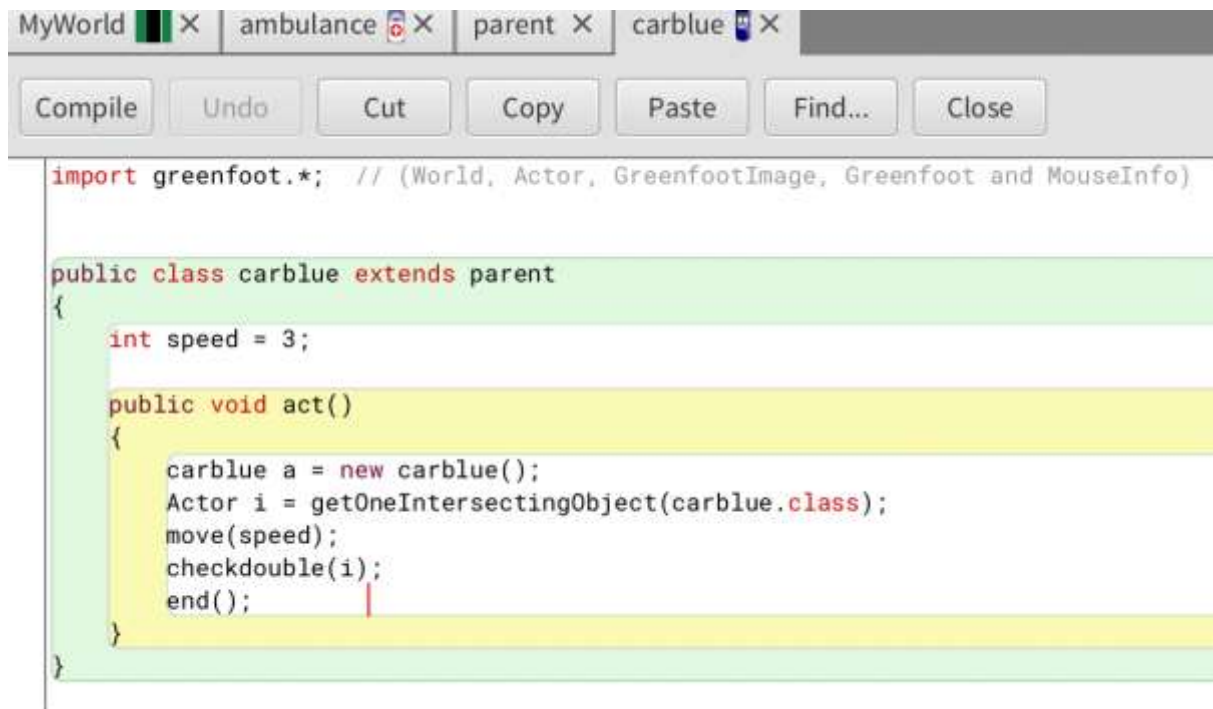
3. Metode Checkdouble:

- Memeriksa apakah terdapat objek i (parameter) yang tidak bernilai null.

- Jika objek *i* tidak bernilai null, objek tersebut dihapus dari dunia menggunakan `getWorld().removeObject(i)`.

4. Metode Move:

- Memindahkan objek parent ke bawah dengan kecepatan yang ditentukan oleh parameter `speed`.
- Mengubah koordinat Y objek sesuai dengan nilai kecepatan.
- Digunakan untuk menggerakkan objek ke bawah dalam simulasi permainan.



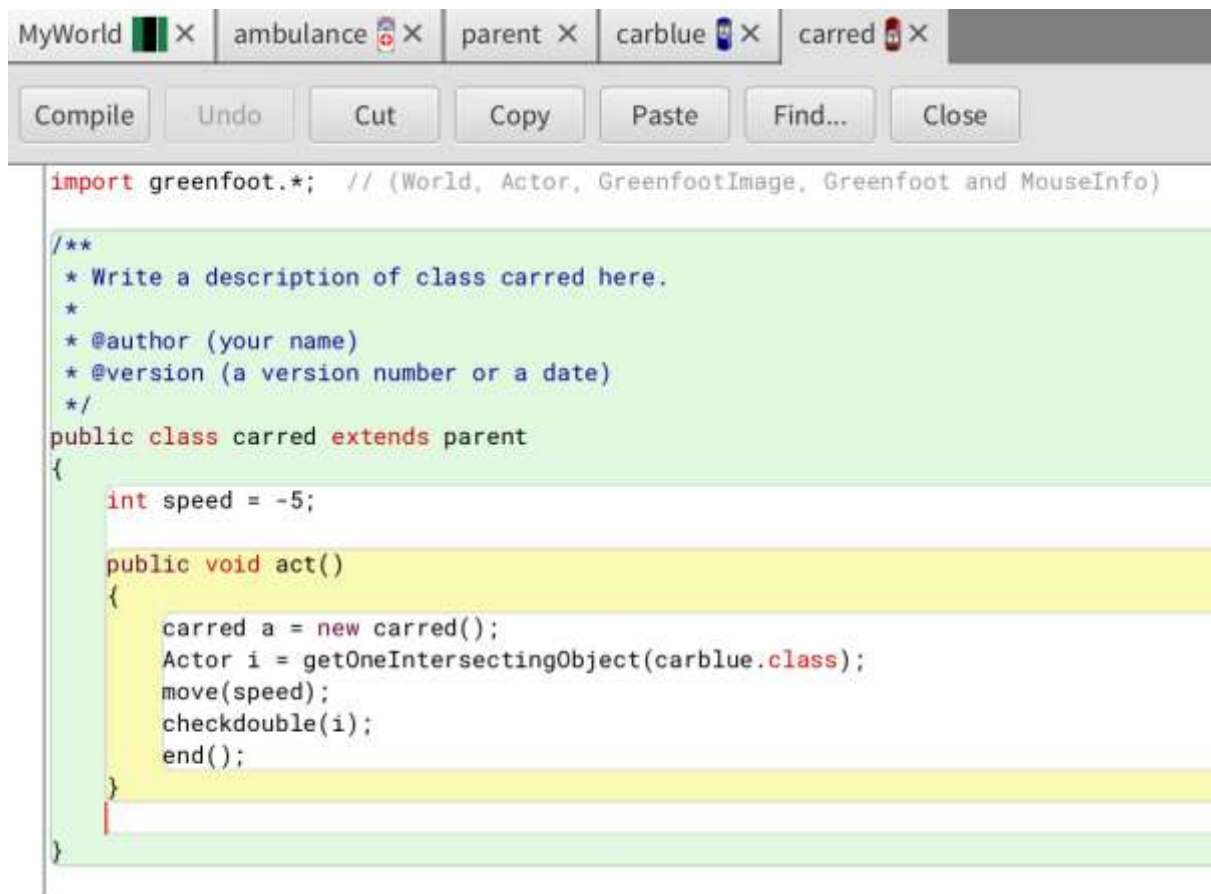
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class carblue extends parent
{
    int speed = 3;

    public void act()
    {
        carblue a = new carblue();
        Actor i = getOneIntersectingObject(carblue.class);
        move(speed);
        checkdouble(i);
        removeObject(i);
    }
}
```

- `int speed = 3;`: Variabel `speed` (int, nilai awal 3) untuk kecepatan pergerakan objek "carblue".
- `public void act() { ... }`: Metode `act` berisi aksi berkala, termasuk pemanggilan metode dari kelas induk dan pemindahan objek.
- `carblue a = new carblue();`: Membuat objek "carblue" baru (tidak terpakai).
- `Actor i = getOneIntersectingObject(carblue.class);`: Mendapatkan objek bersinggungan dengan "carblue" dan menyimpannya di variabel `i`.
- `move(speed);`: Memindahkan "carblue" ke bawah dengan kecepatan dari variabel `speed`.
- `checkdouble(i);`: Memeriksa tumpang tindih dengan objek lain (`i`) dan menghapusnya jika iya.

- end();: Memeriksa apakah "carblue" di luar batas atas atau bawah dunia, kemudian menghapusnya jika perlu.



The screenshot shows a Greenfoot IDE window with several tabs: 'MyWorld', 'ambulance', 'parent', 'carblue', and 'carred'. The 'carred' tab is active, displaying the following Java code:

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

/**
 * Write a description of class carred here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class carred extends parent
{
    int speed = -5;

    public void act()
    {
        carred a = new carred();
        Actor i = getOneIntersectingObject(carblue.class);
        move(speed);
        checkdouble(i);
        end();
    }
}
```

1. Variabel Speed:

- int speed = -5;: Variabel integer speed dideklarasikan dengan nilai awal -5.
- Menentukan kecepatan pergerakan objek carred, dengan nilai negatif menunjukkan pergerakan ke atas.

2. Metode Act:

- public void act() { ... }: Metode act dijalankan berkala selama simulasi berjalan.
- Mencakup beberapa aksi, termasuk pemanggilan metode dari kelas induk dan pemindahan objek.
- carred a = new carred();: Membuat objek carred baru (tampaknya tidak digunakan atau ditambahkan ke dunia).
- Actor i = getOneIntersectingObject(carblue.class);: Mendapatkan objek yang bersinggungan dengan objek carred dan menyimpannya dalam variabel i.

- `move(speed);`: Memanggil metode `move` dari kelas induk untuk memindahkan objek `carred` ke atas dengan kecepatan dari variabel `speed`.
- `checkdouble(i);`: Memanggil metode `checkdouble` dari kelas induk untuk memeriksa apakah objek `carred` bersinggungan dengan objek lain (`i`) dan menghapusnya jika iya.
- `end();`: Memanggil metode `end` dari kelas induk untuk memeriksa apakah objek `carred` berada di luar batas atas atau bawah dunia, dan menghapusnya jika ya.

```

public class coin extends parent
{
    int speed = 3;
    public void act()
    {
        coin a = new coin();
        Actor i = getOneIntersectingObject(coin.class);
        move(speed);
        checkdouble(i);
        end();
    }
}

```

coin:

1. **`int speed = 3;`**: Mendeklarasikan variabel **`speed`** dengan tipe data integer dan memberikan nilai awal 3. Variabel ini digunakan untuk menentukan kecepatan pergerakan objek **`coin`**.
2. **`public void act() { ... }`**: Metode **`act`** yang akan dijalankan secara berkala selama simulasi berjalan. Di dalamnya terdapat beberapa aksi, termasuk pemanggilan metode dari kelas induk dan pemindahan objek.
 - **`coin a = new coin();`**: Membuat objek **`coin`** baru. Namun, tampaknya objek ini tidak digunakan atau ditambahkan ke dunia.
 - **`Actor i = getOneIntersectingObject(coin.class);`**: Mendapatkan objek yang bersinggungan dengan objek **`coin`** saat ini dan menyimpannya dalam variabel **`i`**.

- **move(speed);**: Memanggil metode **move** dari kelas induk (**parent**) untuk memindahkan objek **coin** ke bawah dengan kecepatan yang ditentukan oleh variabel **speed**.
- **checkdouble(i);**: Memanggil metode **checkdouble** dari kelas induk untuk memeriksa apakah objek **coin** bersinggungan dengan objek lain (**i**). Jika ya, objek tersebut dihapus dari dunia.
- **end();**: Memanggil metode **end** dari kelas induk untuk memeriksa apakah objek **coin** berada di luar batas atas atau bawah dunia. Jika ya, objek ini dihapus dari dunia.

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

public class tree extends parent
{
    int speed = 3;
    public void act()
    {
        tree a = new tree();
        Actor i = getOneIntersectingObject(tree.class);
        move(speed);
        checkdouble(i);
        end();
    }
}
```

- **int speed = 3;**: Kecepatan pergerakan objek tree diinisialisasi dengan nilai 3.
- Metode **act()**: Berisi aksi-aksi yang dijalankan secara berkala selama simulasi.
 - **tree a = new tree();**: Objek tree baru dibuat tanpa tampak digunakan atau ditambahkan ke dunia.
 - **Actor i = getOneIntersectingObject(tree.class);**: Mendapatkan objek yang bersinggungan dengan objek tree dan menyimpannya dalam variabel **i**.
 - **move(speed);**: Memindahkan objek tree ke bawah dengan kecepatan dari variabel **speed**.
 - **checkdouble(i);**: Memeriksa dan menghapus objek tree jika bersinggungan dengan objek lain (**i**).
 - **end();**: Memeriksa dan menghapus objek tree jika berada di luar batas atas atau bawah dunia.

