

**UJIAN AKHIR SEMESTER BERBASIS PROYEK MATA KULIAH
PERMROGRAMAN BERORIENTASI OBJEK (PBO)**

DOSEN PENGAMPU : SAYEKTI HARITS SURYAWAN, S.Kom., M.Kom.



Oleh:

NURUL KHAFIFAH HAMRUN

(2211102441078)

ANDI RYAN PRAMADANA PUTRA

(2211102441084)

M. NUR BAHRI

(2211102441122)

**FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR
DESEMBER 2023**

BAB 1

PENDAHULUAN

A. Latar Belakang

Pemrograman Berorientasi Object atau dalam bahasa Inggris lebih dikenal dengan Object Oriented Programming (OOP) adalah sebuah paradigma dalam pemrograman yang menyelesaikan masalah program dengan menyediakan objek-objek (terdiri dari beberapa attribute dan method) yang saling berkaitan dan disusun kedalam satu kelompok atau yang disebut dengan class. Nantinya objek-objek tersebut akan saling berinteraksi untuk menyelesaikan masalah program yang rumit. Pemrograman Berbasis Objek (PBO) digunakan dalam Greenfoot untuk memfasilitasi pengembangan permainan komputer berorientasi objek. Greenfoot dirancang khusus untuk pendidikan dan pengajaran pemrograman, terutama untuk siswa dan mahasiswa yang baru mengenal konsep PBO.

Penggunaan PBO dalam Greenfoot adalah representasi objek dan kelas dimana setiap elemen dalam permainan direpresentasikan sebagai objek yang merupakan instance dari suatu kelas. Kemudian ada pewarisan (inheritance). Yaitu dimana kelas anak dapat mewarisi sifat dan perilaku dari kelas induk. Selanjutnya, polimorfisme. Yaitu polimorfisme memungkinkan pemanggilan metode yang sama untuk objek - objek tersebut dapat memberikan respons yang berbeda. Kemudian, enkapsulasi. Prinsip enkapsulasi melibatkan pembungkusan data dan metode dalam satu unit (kelas), dan hanya memberikan akses terbatas ke data tersebut. Kemudian interaksi antar objek, Dalam Greenfoot, ini dapat dicapai dengan memanggil metode antar objek untuk menginisiasi tindakan atau respons tertentu, seperti deteksi tabrakan atau kerjasama antar elemen. Terakhir yaitu Overriding dan Overloading, Overriding memungkinkan penggantian perilaku metode dari kelas induk, sementara overloading memungkinkan pembuatan metode dengan parameter yang berbeda

Laporan ini bertujuan untuk memberikan pemahaman yang mendalam tentang konsep pemrograman berbasis objek dengan menggunakan platform pengembangan permainan edukatif yang disebut Greenfoot. Konsep-konsep dasar seperti objek dan kelas, interaksi antar objek, enkapsulasi, pewarisan, Overriding dan Overloading dan polimorfisme akan dijelaskan secara terperinci, sambil memberikan panduan langkah-demi-langkah tentang cara

mengimplementasikannya menggunakan Greenfoot. Dalam laporan ini, pembaca akan diajak untuk memahami fitur-fitur utama Greenfoot dan bagaimana platform ini dapat menjadi alat yang efektif untuk pembelajaran pemrograman berbasis objek.

BAB II

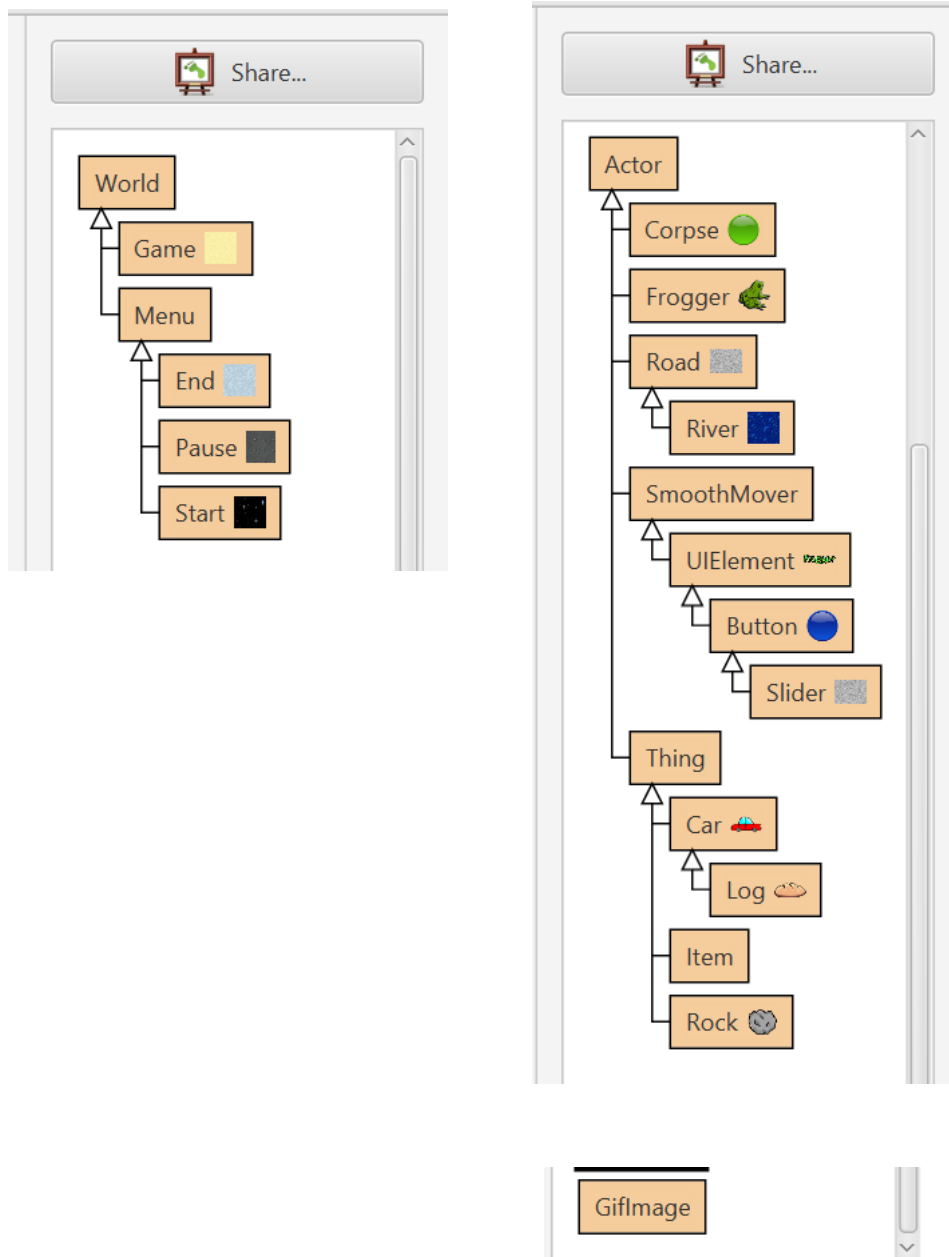
PEMBAHASAN

A. Game dengan Konsep Pemrograman Berorientasi Objek (PBO)/OOP

a. Kelas dan Objek

➤ Kelas (Class)

Kelas adalah blueprint atau template untuk menciptakan objek. Ini menentukan atribut atau perilaku yang dimiliki oleh objek-objek yang dibuat dari kelas tersebut.



➤ Objek

Objek, adalah instansi konkret dari suatu class yang memiliki atribut dan metode yang didefinisikan oleh class tersebut.

Berikut adalah objek yang terdapat pada game:

- “World” terdiri dari “Game” yang lebih lanjut terdiri dari “Menu”, “End”, “Pause”, dan “Start”.
- “Actor” terdiri dari “Corpse”, “Frogger”, “Road”, “River”, “SmoothMover”, dan “UIElement”.
- “Thing” terdiri dari “Car”, “Log”, dan “Rock”.
- “Clickable” terdiri dari “GifImage”.

b. Pewarisan (inheritance)

Inheritance/pewarisan ini maksudnya sifat dalam PBO yang memungkinkan sifat-sifat dari suatu kelas diturunkan ke kelas lainnya. Pewarisan ini bisa dilakukan secara bertingkat, sehingga semakin kebawah maka kelas tersebut semakin spesifik.

Berikut adalah beberapa kelas – kelas yang merupakan pewarisan:

1. Corpse :

```
import greenfoot.*; // (World, Actor, Green
public class Corpse extends Actor
```

Kelas “Corpse” merupakan (subclass) dari kelas “Actor”

2. Frogger

```
import greenfoot.*; // (World, Actor, (
import java.util.ArrayList;

public class Frogger extends Actor
```

Kelas “Frogger” merupakan (subclass) dari kelas “Actor”

3. Road

```
import greenfoot.*; // (World, Actor
import java.util.ArrayList;

public class Road extends Actor
```

Kelas “Road” merupakan (subclass) dari kelas “Actor”

4. River

```
import greenfoot.*; // (World, Actor, G
import java.util.ArrayList;

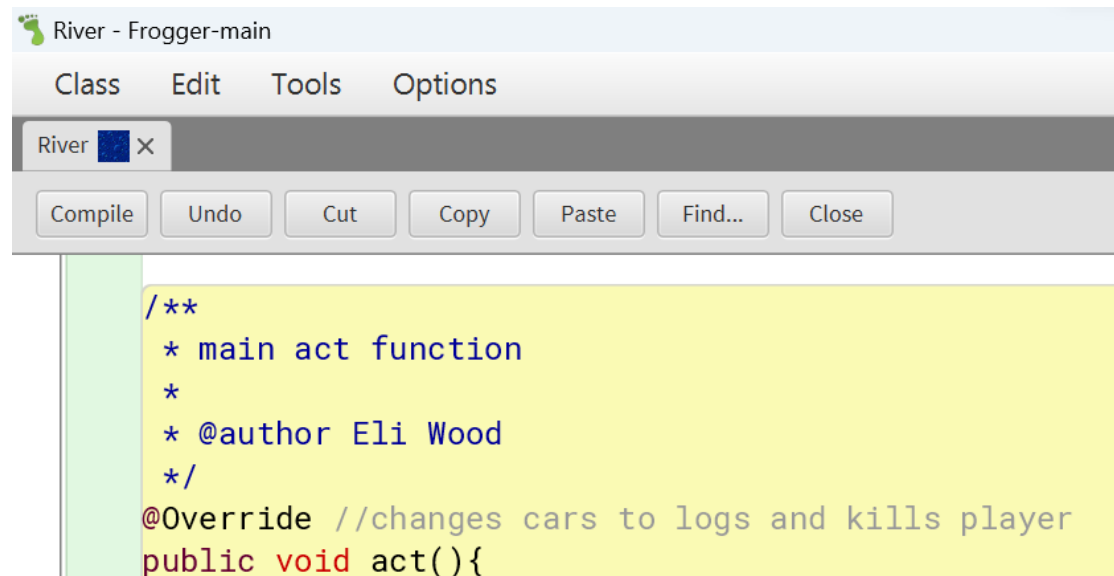
public class River extends Road{
```

Kelas “River” merupakan (subclass) dari kelas “Road”

c. Polimorfisme

Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk” method yang berbeda-beda meskipun namanya sama.

polimorfisme dapat terlihat pada implementasi metode act() dan penggunaan objek Rock dan Log sebagai turunan dari kelas Car dalam kelas River



Pada kelas River, terdapat peng-overridingan metode act() yang diwarisi dari kelas Road. Dalam metode ini, perilaku dari kelas Road diubah untuk memenuhi kebutuhan spesifik kelas River. Hal ini menunjukkan polimorfisme, di mana kelas River memberikan implementasi yang berbeda untuk metode yang sama yang diwarisi dari kelas Road

d. Enkapsulasi

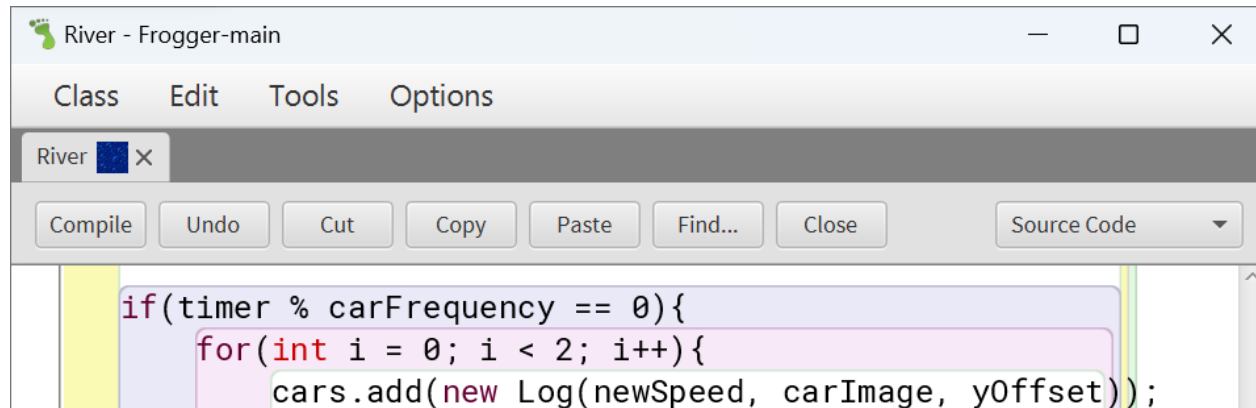
Enkapsulasi adalah salah satu prinsip dalam pemrograman berorientasi objek yang mengacu pada pembungkusan (encapsulation) data dan metode yang beroperasi pada data ke dalam satu unit yang disebut sebagai kelas. Prinsip ini memungkinkan kita untuk menyembunyikan rincian implementasi internal dari objek dan hanya mengekspos fungsionalitas yang diperlukan.

e. Interaksi Antar Objek

Interaksi antar objek dalam pemrograman berorientasi objek merujuk pada hubungan dan pertukaran informasi yang terjadi antara dua atau lebih objek yang bekerja bersama untuk mencapai suatu tujuan atau perilaku tertentu. Pada dasarnya, objek-objek berkomunikasi dan berkoordinasi satu sama lain melalui metode dan properti yang mereka miliki.

Pada metode `act()` kelas `River`, terdapat beberapa interaksi antar objek:

1. Penambahan Object Log ke dalam ArrayList `cars`:



```
if(timer % carFrequency == 0){
    for(int i = 0; i < 2; i++){
        cars.add(new Log(newSpeed, carImage, yOffset));
    }
}
```

2. Pemanggilan metode `moveTo` pada objek 'Log' dan 'Rock':

```

for(int i = 0; i < rocks.size(); i++){
    rocks.get(i).moveTo(rocks.get(i).getX(), y);
    if(rocks.get(i).getDead()){
        world.removeObject(rocks.get(i));
        rocks.remove(i);
        i--;
    }

    if(rocks.get(i).touchingFrog()){
        kill = false;
    }
}

```

```

for(int i = 0; i < cars.size(); i++){
    cars.get(i).moveTo(cars.get(i).getX(), y);

    if(cars.get(i).getDead()){
        world.removeObject(cars.get(i));
        cars.remove(i);
        i--;
    }
}

```

3. Pengecekan tabrakan dengan objek 'Frog'

```

if(intersects(world.getFrog()) && kill){
    world.getFrog().drown();
}

```

4. Pemanggilan Metode drown() pada objek 'Frog':

```

if(intersects(world.getFrog()) && kill){
    world.getFrog().drown();
}

```

Semua interaksi ini mengindikasikan bagaimana objek River berinteraksi dengan objek-objek 'Log', 'Rock', 'Frog' di dalamnya

f. Overriding dan Overloading

➤ Overriding:

1. Metode checkCollision() pada Kelas Car dan Log:
 - Kelas Log melakukan overriding terhadap metode checkCollision() yang diwarisi dari kelas induk Car
 - .Overriding dilakukan untuk menyesuaikan perilaku deteksi tabrakan dengan objek Frog.
2. Metode act() pada Kelas Rock:
 - Kelas Rock melakukan overriding terhadap metode act() yang diwarisi dari kelas induk Thing.
 - Overriding dilakukan untuk menyesuaikan perilaku aksi yang terjadi saat act pada objek Rock.

➤ Overloading:

3. Constructor Kelas Car:
 - Kelas Car memiliki beberapa konstruktor yang menunjukkan overloading. Beberapa konstruktor tersebut memiliki jumlah parameter dan tipe parameter yang berbeda.
4. Metode moveTo() pada Kelas Car dan Log:
 - Kelas Car dan Log memiliki metode moveTo(). Terdapat beberapa versi metode moveTo() yang menunjukkan overloading karena berbeda dalam jumlah dan tipe parameter.
5. Metode checkEdge() pada Kelas Car dan Rock:
 - Kelas Car dan Rock memiliki metode checkEdge() yang menunjukkan overloading karena keduanya memiliki metode dengan nama yang sama tetapi menerima jumlah dan tipe parameter yang berbeda.
6. Constructor Kelas Item:
 - Kelas Item memiliki beberapa konstruktor yang menunjukkan overloading. Beberapa konstruktor tersebut memiliki jumlah parameter dan tipe parameter yang berbeda.
7. Metode moveTo() pada Kelas Item:
 - Kelas Item memiliki metode moveTo() yang menunjukkan overloading karena memiliki beberapa versi dengan jumlah dan tipe parameter yang berbeda.

BAB III

PENUTUP

Kesimpulan :

Laporan ini menjelaskan konsep Pemrograman Berbasis Objek (OOP) dengan menggunakan platform Greenfoot. Beberapa konsep dasar PBO, seperti objek dan kelas, pewarisan, polimorfisme, enkapsulasi, interaksi antar objek, overriding, dan overloading, diuraikan dengan contoh implementasi dalam pembuatan permainan. Laporan ini memberikan wawasan menyeluruh tentang konsep PBO dan penerapannya dalam Greenfoot, menjadikannya alat pembelajaran efektif.