LAPORAN PROJECT UAS PBO / OOP

INF2143/2153

Game Semut Sederhana Greenfoot

Oleh :

Dhimas Arief Kurniawan / 2211102441171

Arya Ikhwanun Sholih / 2211102441228

Henri / 2211102441161

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

Membuat sebuah game menggunakan platform Greenfoot yang menunjukkan penerapan konsep-konsep Pemrograman Berbasis Objek. Kali ini, kami akan bahas pembuatan game sederhana ini per tipe/kelas Berikut ini adalah kelas kelas yang ada dalam game ini:

1. **Actor Cicak (Cicak):**
    - Kelas ini merepresentasikan aktor cicak dalam game Anda.
    - Mungkin memiliki metode untuk bergerak, menanggapi input pemain, dan berinteraksi dengan objek lain di lingkungan game.
2. **Actor Semut (Semut):**
    - Kelas ini merepresentasikan aktor semut dalam game.
    - Sama seperti kelas cicak, dapat memiliki metode untuk bergerak, berinteraksi, dan mungkin memiliki perilaku khusus semut.
3. **Actor Anim1 (Animasi Semut Pada Menu Home):**
    - Kelas ini merepresentasikan aktor semut dalam game.
    - Hanya berupa animasi yang ditampilkan ketika memasuki game yang terdapat pada menu home.
4. **Tampilan About (AboutScreen):**
    - Kelas ini mewakili tampilan "About" di game Anda.
    - Mungkin berisi informasi tentang pembuat game, petunjuk, atau deskripsi singkat tentang permainan.
5. **Tombol Keluar (ExitButton):**
    - Kelas ini merepresentasikan tombol keluar pada menu atau tampilan lainnya.
    - Mungkin memiliki metode untuk menangani peristiwa klik dan keluar dari permainan.
6. **Tampilan Menu (MenuScreen):**
    - Kelas ini mewakili tampilan menu utama permainan.
    - Mungkin berisi tombol-tombol untuk mulai permainan, melihat tampilan "About", dan keluar dari permainan.
7. **Tombol Mulai (StartButton):**
    - Kelas ini merepresentasikan tombol mulai pada menu.
    - Mungkin memiliki metode untuk memulai permainan atau mengalihkan ke layar permainan utama.
8. **Objek Pizza (Pizza):**
    - Kelas ini merepresentasikan objek pizza dalam game.
    - Mungkin digunakan sebagai objek yang harus dikumpulkan atau dihindari oleh aktor seperti cicak, semut, dan lebah.

9. **Tombol Restart (RestartButton):**
    - Kelas ini merepresentasikan tombol restart yang dapat digunakan untuk memulai ulang permainan.
    - Mungkin memiliki metode untuk mereset posisi atau keadaan permainan ke kondisi awal.

Setiap kelas memiliki atribut dan metode yang sesuai dengan fungsinya masing-masing dalam game. Dengan pengorganisasian yang baik, dapat membuat game sederhana yang menarik dan menyenangkan dengan Greenfoot.

Kelas dunia (world class) yang dapat diterapkan untuk mengatur latar belakang atau tampilan dalam game Anda:

1. **Dunia Home (HomeWorld):**
    - Kelas ini merepresentasikan dunia atau latar belakang ketika pemain berada di layar utama atau layar rumah (home).
    - Mungkin berisi elemen-elemen seperti gambar latar belakang, tombol-tombol menu, dan elemen-elemen lain yang terlihat di layar utama.
2. **Dunia Lose (LoseWorld):**
    - Kelas ini merepresentasikan dunia atau latar belakang ketika pemain kalah dalam permainan.
    - Mungkin berisi elemen-elemen seperti pesan kekalahan, gambar atau animasi kalah, dan opsi untuk mencoba lagi atau kembali ke layar utama.
3. **Dunia Win (WinWorld):**
    - Kelas ini merepresentasikan dunia atau latar belakang ketika pemain menang dalam permainan.
    - Mungkin berisi elemen-elemen seperti pesan kemenangan, gambar atau animasi kemenangan, dan opsi untuk melanjutkan ke level berikutnya atau kembali ke layar utama.
4. **Dunia About (AboutWorld):**
    - Kelas ini merepresentasikan dunia atau latar belakang ketika pemain melihat informasi "About" dalam permainan.
    - Mungkin berisi elemen-elemen seperti teks penjelasan, gambar pembuat game, dan tombol-tombol untuk kembali ke layar utama.
5. **Dunia Tema MyWorld (MyWorld):**
    - Kelas ini dapat digunakan untuk merepresentasikan dunia atau latar belakang utama permainan.
    - Mungkin menjadi tempat pemain berinteraksi dengan aktor-aktor seperti cicak, semut, lebah, dan objek-objek seperti pizza.

- Berisi elemen-elemen seperti gambar latar belakang, elemen-elemen permainan, dan mungkin beberapa objek pengaturan lainnya.

Kemudian untuk Inheritance adalah konsep dasar dalam pemrograman berorientasi objek yang memungkinkan kelas baru (kelas anak atau subclass) mewarisi sifat dan perilaku dari kelas yang sudah ada (kelas induk atau superclass). Berikut adalah konsep inheritance yang dapat diterapkan pada kelas-kelas dalam game Greenfoot yang dibuat:

1. **Actor Cicak (Cicak), Actor Semut (Semut), dan Actor Lebah (Lebah):**
    - Ketiga kelas ini dapat mewarisi sifat dari kelas umum, misalnya, **AnimalActor** atau **InsectActor**. Kelas induk ini bisa berisi metode dan atribut yang umum untuk semua aktor hewan atau serangga dalam permainan, seperti metode **bergerak()**, **interaksi()**, dan atribut **kecepatan**.
2. **Tampilan About (AboutScreen), Tombol Keluar (ExitButton), Tampilan Menu (MenuScreen), Tombol Mulai (StartButton), Tombol Restart (RestartButton):**
    - Kelas-kelas ini dapat mewarisi dari kelas **ScreenElement** atau **ButtonElement**. Kelas induk ini bisa mengandung metode dan atribut yang umum untuk elemen-elemen layar atau tombol dalam permainan, seperti metode **tampilkan()**, **sembunyikan()**, dan atribut **warnaLatar**.
3. **Objek Pizza (Pizza):**
    - Kelas ini dapat mewarisi dari kelas **GameObject**. Kelas induk ini bisa memiliki metode dan atribut yang umum untuk objek-objek dalam permainan, seperti metode **gerakRandom()**, **hilang()**, dan atribut **skor**.
4. **Dunia Home (HomeWorld), Dunia Lose (LoseWorld), Dunia Win (WinWorld), Dunia About (AboutWorld), Dunia Tema MyWorld (MyWorld):**
    - Kelas-kelas dunia ini dapat mewarisi dari kelas **GreenfootWorld** atau kelas lain yang menyediakan fungsionalitas dunia dalam permainan. Kelas induk ini dapat berisi metode dan atribut yang umum untuk manajemen dunia, seperti metode **aturAktivitas()**, **gantiDunia()**, dan atribut **backgroundImage**.

Dengan menerapkan konsep inheritance ini dapat mengurangi redundansi dalam kode, meningkatkan keterbacaan, dan memudahkan pemeliharaan. Misalnya, jika ada perubahan yang perlu dilakukan pada semua elemen layar, hanya perlu memodifikasi kelas **ScreenElement** atau **ButtonElement**, dan perubahan tersebut akan secara otomatis berdampak pada semua kelas anak yang mewarisinya.

Polimorfisme adalah konsep dalam pemrograman berorientasi objek yang memungkinkan objek untuk menunjukkan perilaku yang sesuai dengan tipe mereka, bahkan jika mereka dipanggil melalui antarmuka yang sama. Dalam konteks Greenfoot, polimorfisme dapat diterapkan melalui metode yang diimplementasikan oleh kelas-kelas yang berbeda meskipun mereka memiliki nama yang sama. Berikut adalah beberapa contoh penerapan polimorfisme pada kelas-kelas dalam game Greenfoot:

1. **Polimorfisme pada Aktor (Cicak, Semut, Lebah):**

- Meskipun ketiga kelas ini mungkin memiliki metode dengan nama yang sama, seperti **bergerak()**, setiap kelas bisa mengimplementasikan metode tersebut dengan perilaku yang berbeda. Sebagai contoh, cicak mungkin bergerak dengan melompat, semut dengan merayap, dan lebah dengan terbang. Pemanggilan metode **bergerak()** pada objek dari kelas apa pun akan memberikan hasil yang sesuai dengan tipe aktor tersebut.

2. **Polimorfisme pada Tombol (StartButton, ExitButton, RestartButton):**
    - Jika ketiga kelas ini memiliki metode yang sama, misalnya, **klik()**, masing-masing kelas bisa mengimplementasikan metode tersebut untuk menangani peristiwa klik dengan cara yang berbeda. Meskipun Anda memanggil metode **klik()** pada objek tombol apa pun, perilaku yang dijalankan akan bergantung pada implementasi kelas tersebut.
3. **Polimorfisme pada Dunia (HomeWorld, LoseWorld, WinWorld, AboutWorld, MyWorld):**
    - Ketika Anda memanggil metode tertentu, seperti **aturAktivitas()**, pada objek dunia, masing-masing dunia dapat memberikan implementasi yang sesuai dengan kebutuhan spesifiknya. Misalnya, dunia yang menang (**WinWorld**) mungkin memanggil metode untuk menampilkan pesan kemenangan, sementara dunia yang kalah (**LoseWorld**) dapat menampilkan pesan kekalahan.

Enkapsulasi adalah salah satu prinsip dasar dalam pemrograman berorientasi objek yang menggabungkan data dan metode yang bekerja pada data tersebut dalam satu unit yang disebut kelas. Enkapsulasi membantu menyembunyikan implementasi internal suatu kelas dan hanya mengekspos antarmuka yang diperlukan ke luar. Dengan kata lain, enkapsulasi melibatkan penyembunyian rincian implementasi internal dari pengguna objek tersebut. Berikut adalah contoh enkapsulasi dalam beberapa kelas dalam game Greenfoot:

1. **Aktor (Cicak, Semut, Lebah):**

Atribut-atribut seperti posisi, kecepatan, dan status internal lainnya yang mungkin diperlukan oleh aktor disimpan secara pribadi sebagai variabel anggota di dalam kelas. Akses ke atribut-atribut ini dibatasi oleh metode-metode publik seperti aturPosisi(), ambilPosisi(), aturKecepatan(), dan ambilKecepatan(). Dengan cara ini, implementasi internal dari bagaimana aktor menyimpan dan memanipulasi data tersebut disembunyikan dari pengguna luar.

2. **Objek Pizza (Pizza):**

Atribut-atribut seperti nilai skor dan status objek pizza disimpan secara privat di dalam kelas. Metode-metode seperti tambahSkor() dan ambilStatus() memberikan antarmuka untuk berinteraksi dengan data ini. Pengguna objek pizza tidak perlu mengetahui bagaimana data diatur atau diolah, tetapi hanya perlu menggunakan metode yang telah disediakan.

3. **Dunia Tema MyWorld (MyWorld):**

   Atribut-atribut yang mencakup elemen-elemen permainan, aktor-aktor, dan data dunia lainnya disimpan secara privat di dalam kelas. Metode-metode seperti tambahAktor(), hapusAktor(), dan perbaruiSkor() memberikan antarmuka yang diperlukan bagi objek luar untuk berinteraksi dengan dunia, sementara detil implementasi internalnya tetap tersembunyi.

Interaksi antar objek adalah konsep penting dalam pemrograman berorientasi objek di mana objek-objek saling berkomunikasi dan berinteraksi satu sama lain untuk mencapai tujuan tertentu. Dalam konteks game Greenfoot, interaksi antar objek dapat mencakup kolaborasi, pertukaran informasi, dan bahkan tindakan yang dilakukan satu objek terhadap objek lainnya. Berikut adalah beberapa contoh interaksi antar objek dalam kelas-kelas yang telah dijelaskan sebelumnya:

1. **Interaksi antara Aktor (Cicak dan Semut) :**
   - Ketika aktor mendekati objek pizza, bisa ada metode **interaksi()** yang dipanggil untuk menangani peristiwa "memakan" pizza. Dalam hal ini, aktor dapat memanggil metode pada objek pizza untuk mengubah skor dan status objek pizza.

```
public void act()
  {
    move(3);
    if (Greenfoot.getRandomNumber(100) < 10) {
       turn(Greenfoot.getRandomNumber(15) - 30);
    }
    makanSemut();
  }
```

2. **Interaksi antara Tombol (StartButton, StoppedButton, HomeButton) dan Dunia (HomeWorld, LoseWorld, WinWorld, AboutWorld, MyWorld):**
   - Ketika tombol ditekan, bisa ada metode **aksiTombol()** yang memanggil metode pada dunia untuk mengubah kondisi permainan atau memindahkan pemain ke dunia lain.

```
public void started()
  {
    music.setVolume(50);
    music.play();
  }

  public void stopped()
  {
    music.stop();
```

```
    music.pause();
  }

  public Home()
  {
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(600, 400, 1);
    prepare();
  }
```

3. **Interaksi antara Dunia (MyWorld) dan Objek Pizza (Pizza):**
    - Ketika objek pizza dibuat atau dimakan, dunia dapat memperbarui skor atau memutuskan untuk menambahkan objek pizza baru ke dunia.

```
private void prepare()
  {
    mulai mulai = new mulai();
    addObject(mulai,86,82);
    keluar keluar = new keluar();
    addObject(keluar,89,311);
    about about = new about();
    addObject(about,88,192);
    anim1 anim1 = new anim1();
    addObject(anim1,188,346);
  }
```

Interaksi antar objek memungkinkan objek-objek dalam game Anda untuk berkolaborasi dan menciptakan perilaku yang lebih kompleks dan dinamis. Dengan merancang metode dan antarmuka dengan baik, Anda dapat mengatur interaksi ini secara efisien dan membuat permainan yang lebih menarik dan interaktif.

Overriding dan overloading adalah dua konsep yang terkait, tetapi memiliki penggunaan yang berbeda dalam pemrograman berorientasi objek.

Overriding:

Overriding terjadi ketika kelas anak (subclass) menyediakan implementasi ulang dari metode yang sudah didefinisikan di kelas induk (superclass). Metode di kelas anak memiliki nama, parameter, dan tipe kembalian yang sama dengan metode di kelas induk.

Contoh Overriding pada kelas **Cicak** (kelas anak dari **Actor** di Greenfoot):

```
public void act()
    {
        move(3);
        if (Greenfoot.getRandomNumber(100) < 10) {
            turn(Greenfoot.getRandomNumber(15) - 30);
        }
        makanSemut();
    }

    /**
     *
     */
    public void makanSemut()
    {
        if (isTouching(Semut.class)) {
            removeTouching(Semut.class);
            Greenfoot.setWorld(new Lose());
        }
    }
```

Dalam contoh ini, metode **act()** di kelas **Cicak** menggantikan metode **act()** yang sudah ada di kelas **Actor**.

Overloading:

Overloading terjadi ketika dua atau lebih metode dalam suatu kelas memiliki nama yang sama tetapi parameter yang berbeda (jumlah, tipe, atau urutan parameter). Dengan kata lain, metode-overloading memungkinkan kita untuk mendefinisikan beberapa versi dari metode dengan nama yang sama dalam suatu kelas.

Contoh Overloading pada kelas **Pizza**:

```
public class pizza extends Actor
{

    /**
     * Act - do whatever the pizza wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
     */
```

```
    public void act()
    {
    }
}
```

Dalam contoh ini, terdapat tiga metode **tambahTopping()** dengan nama yang sama, tetapi parameter yang berbeda.

SELESAI !!!

Semoga proyek ini memberikan kesempatan untuk mendalami dan menerapkan konsep-konsep yang telah dipelajari, sekian trimakasih.