PEMROGRAMAN BERORIENTASI OBJEK

INF2143/53

LAPORAN PROJECT UAS:

Green Field

Oleh :

Nama Anggota Kelompok 1:

1. *Rizky Lukman Haidi (2211102441153)*
2. *Naufal Rajwa Dwiyana Athallah (2211102441136)*
3. *Muhammad Iryadudin Islami (2211102441128)*

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

Laporan Project UAS:

Latar Belakang

Pemrograman Berorientasi Objek (PBO) adalah paradigma pemrograman yang berfokus pada konsep objek. Paradigma ini menggambarkan program sebagai set objek yang berinteraksi satu sama lain untuk mencapai tujuan tertentu. Dalam konteks ini, objek adalah representasi dari entitas atau konsep nyata yang ada di dunia nyata yang memiliki karakteristik (data) dan metode (fungsi atau perilaku).

Tujuan

Tujuan pembuatan game Greenfoot dengan tema Farming Simulator yang ga farming simulator banget pada proyek UAS dapat mencakup beberapa hal, seperti belajar pemrograman, mendesain permainan, dan memahami konsep PBO.

1. Pengembangan Keterampilan Pemrograman :
    - Memahami konsep Pemrograman Berorientasi Objek (PBO) dan menggunakannya saat membuat game.
    - Menerapkan konsep dasar seperti kelas, objek, metode, dan atribut digunakan saat menjalankan game.

2. Pemahaman Greenfoot :
    - Memahami dan menggunakan elemen utama Greenfoot, seperti aktor, dunia, dan teknik yang ditawarkan oleh lingkungan pemrograman.
    - Mengambil keuntungan dari Greenfoot untuk membantu dalam pengembangan permainan berbasis objek.

3. Interaksi dan Pengelolaan Objek :
    - Mengimplementasikan interaksi antara objek-objek di dalam permainan, seperti pertumbuhan tanaman, pengumpulan hasil pertanian, dan pengelolaan sumber daya.
    - Menyertakan elemen manajemen sumber daya seperti uang, energi, dan waktu dalam permainan, pemain harus membuat keputusan strategis untuk meningkatkan hasil pertanian mereka.

Kelas

Dalam pemrograman berorientasi objek (PBO), kelas adalah sebuah blueprint atau cetakan untuk membuat objek. Kelas mendefinisikan atribut (variabel) dan metode (fungsi) yang akan dimiliki oleh objek yang dibuat berdasarkan kelas tersebut.

Di dalam kelas, Anda bisa mendefinisikan perilaku objek dan bagaimana objek berinteraksi dengan data. Contohnya, jika Anda membuat kelas "Mobil", Anda bisa mendefinisikan atribut seperti merek, model, dan metode seperti "start_engine()" atau "stop_engine()".

Setiap objek yang dibuat dari kelas tersebut memiliki akses ke atribut dan metode yang telah didefinisikan dalam kelas tersebut. Hal ini memungkinkan untuk menciptakan banyak objek yang memiliki karakteristik yang sama, tetapi nilai atribut yang berbeda-beda.

Dalam PBO, kelas memainkan peran penting dalam membangun struktur program yang terorganisir dan memfasilitasi konsep pewarisan, enkapsulasi, dan polimorfisme untuk mempermudah pengembangan dan pemeliharaan kode.

Objek

Dalam pemrograman berorientasi objek (PBO), objek adalah representasi konkret dari sebuah kelas. Objek diciptakan berdasarkan blueprint atau definisi yang diberikan oleh kelas. Setiap objek memiliki atribut (variabel) dan metode (fungsi) yang telah didefinisikan dalam kelas.

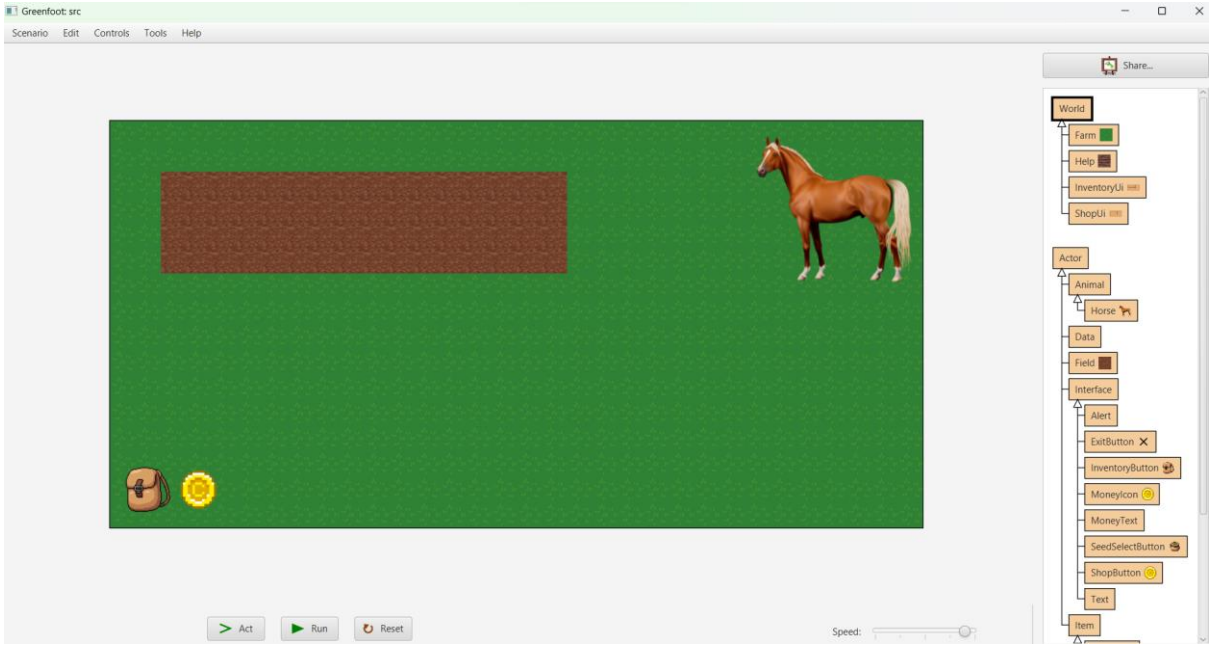
Misalnya, jika Anda memiliki kelas "Mobil", objek dari kelas tersebut bisa menjadi mobil dengan atribut seperti merek, model, warna, dan metode seperti "start_engine()" atau "stop_engine()". Setiap objek mobil yang diciptakan dari kelas tersebut akan memiliki nilai-nilai spesifik untuk atribut-atributnya sendiri.

Objek dalam PBO memungkinkan Anda untuk mengelompokkan data dan fungsionalitas yang terkait bersama-sama sehingga mempermudah pemrograman dan pengelolaan kode. Objek dapat berinteraksi satu sama lain melalui metode-metode yang ada dalam kelas, memungkinkan program untuk melakukan tugas-tugas yang kompleks dengan cara yang terstruktur dan terorganisir.

Pengerjaan Projek UAS:

Berikut merupakan proses pembuatan projek uas yang bernama Green Field, yang menampilkan source code dan hasilnya

1. Gambar berikut menampilkan tampilan awal dari game “Green Field”



2. Gambar berikut merupakan tampilan dari code dari kelas “Farm”

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

/**
 * Main world of this project.
 */
public class Farm extends World
{
    // Membuat lahan untuk ditanami
    int[][] fields = {
            {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
            {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
            {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
            {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
        };
    //koordinat awal penempatan lapangan (pojok kiri/atas)
    int[] fieldsStartPos = new int[]{2,2};

    public Farm()
    {
        // membuat new world dengan sel 32x16 dengan ukuran sel 32x32 piksel.
        super(32, 16, 32);
        prepare();

        // memberikan player uang pada new save
        Data data = new Data();
        if(data.getMoney()<=0){
            data.setMoney(20);
        }
    }
```

```
    public void act(){
        // mengatur agar kecepatan tumbuh tumbuhan tidak dapat di atur player
        Greenfoot.setSpeed(50);

        if(Greenfoot.isKeyDown("f1")) {
            Greenfoot.setWorld(new Help(this));
        }
    }

    private void prepare()
    {
        //menambahkan objek kuda, invetory dan shop kedalam dunia
        addObject(new InventoryButton(),1,14);
        addObject(new ShopButton(),3,14);
        addObject(new Horse(),28,3);

        // menaruh lada pada dunia game
        for (int i = 0; i < this.fields.length; i++) {
            for (int j = 0; j < this.fields[i].length; j++) {
                if(this.fields[i][j] == 1) {
                    Field field = new Field();
                    addObject(field, this.fieldsStartPos[0]+j, this.fieldsStartPos[1]+i);
                }
            }
        }
    }
}
```

Pada gambar diatas menampilkan sebua kelas yang bernama “Farm” yang mana kelas ini merupakan subclass dari kelas Greenfoot "World". Kelas "Farm" ini adalah dunia utama (world) dari proyek yang sedang dibuat.

3. Gambar berikut merupakan tampilan dari code dari kelas “Help”

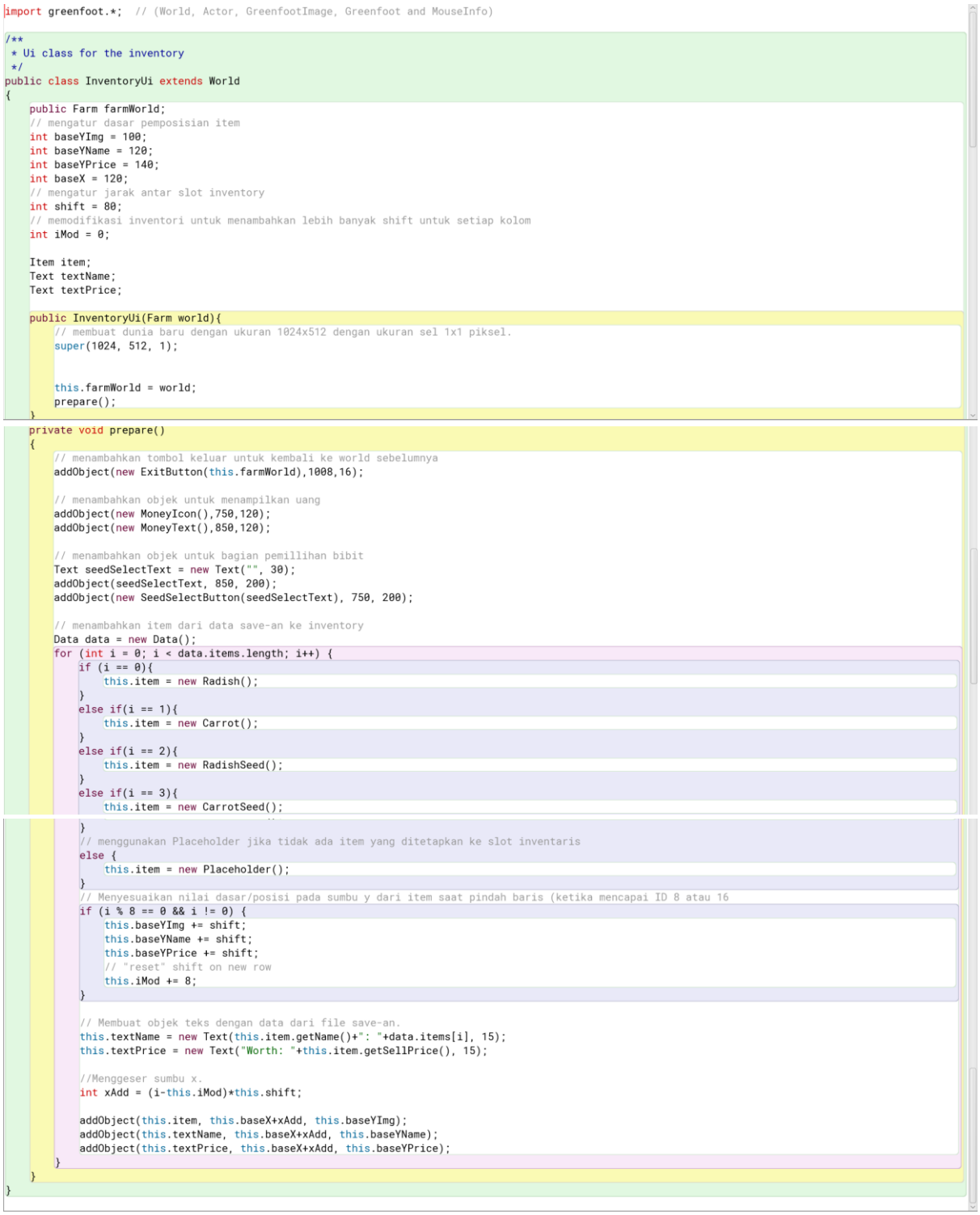
```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Class to show help.
 */
public class Help extends World
{
    World farmWorld;
    //menambahkan kelas help untuk memberikan tutorial cara bermain
    public Help(Farm world)
    {
        //membuat dunia baru dengan sel 32x16 dengan ukuran sel 32x32 piksel.

        super(32, 16, 32);

        //inisialisasi kelas "help" dengan nilai parameter world agar memungkinkan untuk kelas "help" objek world
        this.farmWorld = world;
        prepare();
    }
```

```
    private void prepare()
    {
        // menambahkan tombol exit untuk kembali ke world sebelumnya
        addObject(new ExitButton(this.farmWorld),1008,16);

        // menambahkan text pada kelas help yang akan ditampilkan sebagai tutorial
        addObject(new Text("Tampilan UI", 40), 16,0);
        addObject(new Text("Tas pada farm = Inventory", 20), 16,1);
        addObject(new Text("koin pada Farm = Shop", 20), 16,2);
        addObject(new Text("icon yang dapat di klik pada UI farm adalah Koin dan Tas", 20), 16,3);

        addObject(new Text("Bertani", 40), 16,4);
        addObject(new Text("klik kanan kiri mouse untuk menanam", 20), 16,5);
        addObject(new Text("klik kanan pada mouse untuk memetik", 20), 16,6);
        addObject(new Text("mengganti jenis bibit (klik kantong bibit)", 20), 16,7);
        addObject(new Text("anda bisa membeli bibit di shop dengan harga yang tertera", 20), 16,8);

        addObject(new Text("berbelanja", 40), 16,9);
        addObject(new Text("untuk membeli item di toko klik pada ikon yang ingin dibeli", 20), 16,10);

        addObject(new Text("menjual", 40), 16,11);
        addObject(new Text("untuk menjual klik item pada inventory", 20), 16,12);
    }
}
```

Gambar diatas menunjukan kelas “Help” yanga mana merupakan subclass dari kelas Greenfoot "World". Tujuan dari kelas ini adalah untuk menampilkan panduan atau bantuan tentang cara bermain dalam dunia permainan. kelas “Help” digunakan untuk memberikan panduan atau tutorial kepada pemain tentang cara menggunakan antarmuka, berinteraksi dengan dunia permainan, dan melakukan berbagai tindakan di dalamnya seperti bertani,

berbelanja, dan menjual barang. Sebagai bagian dari permainan, kelas ini memberikan informasi yang berguna kepada pemain untuk memahami cara bermain dengan lebih baik.

4. Gambar berikut merupakan tampilan dari code dari kelas “InventroyUi”



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

/**
 * Ui class for the inventory
 */
public class InventoryUi extends World
{
    public Farm farmWorld;
    // mengatur dasar pemposisian item
    int baseYImg = 100;
    int baseYName = 120;
    int baseYPrice = 140;
    int baseX = 120;
    // mengatur jarak antar slot inventory
    int shift = 80;
    // memodifikasi inventori untuk menambahkan lebih banyak shift untuk setiap kolom
    int iMod = 0;

    Item item;
    Text textName;
    Text textPrice;

    public InventoryUi(Farm world){
        // membuat dunia baru dengan ukuran 1024x512 dengan ukuran sel 1x1 piksel.
        super(1024, 512, 1);


        this.farmWorld = world;
        prepare();
    }
    private void prepare()
    {
        // menambahkan tombol keluar untuk kembali ke world sebelumnya
        addObject(new ExitButton(this.farmWorld),1008,16);

        // menambahkan objek untuk menampilkan uang
        addObject(new MoneyIcon(),750,120);
        addObject(new MoneyText(),850,120);

        // menambahkan objek untuk bagian pemillihan bibit
        Text seedSelectText = new Text("", 30);
        addObject(seedSelectText, 850, 200);
        addObject(new SeedSelectButton(seedSelectText), 750, 200);

        // menambahkan item dari data save-an ke inventory
        Data data = new Data();
        for (int i = 0; i < data.items.length; i++) {
            if (i == 0){
                this.item = new Radish();
            }
            else if(i == 1){
                this.item = new Carrot();
            }
            else if(i == 2){
                this.item = new RadishSeed();
            }
            else if(i == 3){
                this.item = new CarrotSeed();
            }
            // menggunakan Placeholder jika tidak ada item yang ditetapkan ke slot inventaris
            else {
                this.item = new Placeholder();
            }
            // Menyesuaikan nilai dasar/posisi pada sumbu y dari item saat pindah baris (ketika mencapai ID 8 atau 16
            if (i % 8 == 0 && i != 0) {
                this.baseYImg += shift;
                this.baseYName += shift;
                this.baseYPrice += shift;
                // "reset" shift on new row
                this.iMod += 8;
            }

            // Membuat objek teks dengan data dari file save-an.
            this.textName = new Text(this.item.getName()+": "+data.items[i], 15);
            this.textPrice = new Text("Worth: "+this.item.getSellPrice(), 15);

            //Menggeser sumbu x.
            int xAdd = (i-this.iMod)*this.shift;

            addObject(this.item, this.baseX+xAdd, this.baseYImg);
            addObject(this.textName, this.baseX+xAdd, this.baseYName);
            addObject(this.textPrice, this.baseX+xAdd, this.baseYPrice);
        }
    }
}
```

Pada kelas “InventoryUi” yang juga merupakan subclass dari kelas Greenfoot "World". Tujuan dari kelas ini adalah untuk menampilkan antarmuka pengguna (UI) untuk inventaris

dalam dunia permainan. kelas “InventoryUi” bertanggung jawab untuk menampilkan inventaris dalam dunia permainan, menunjukkan item-item yang dimiliki, menampilkan teks informasi tentang setiap item, dan memungkinkan interaksi terkait inventaris kepada pemain. Ini memungkinkan pemain untuk melihat inventaris mereka dan melakukan tindakan seperti menjual atau menggunakan item dalam permainan.

5. Gambar berikut merupakan tampilan dari code dari kelas “ShopUi”

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

/**
 * Ui class for the shop.
 */
public class ShopUi extends World
{
    public Farm farmWorld;
    // nilai-nilai posisi dasar untuk item
    int baseYImg = 100;
    int baseYName = 120;
    int baseYPrice = 140;
    int baseX = 120;
    // mengatur jarak antar slot inventory
    int shift = 80;
    // memodifikasi inventori untuk menambahkan lebih banyak shift untuk setiap kolom
    int iMod = 0;

    Item item;
    Text textName;
    Text textPrice;

    public ShopUi(Farm world){
        // membuat dunia baru dengan ukuran 1024x512 dengan ukuran sel 1x1 piksel.
        super(1024, 512, 1);


        this.farmWorld = world;
        prepare();
    }

    private void prepare(){
        //menambahkan tombol exit
        addObject(new ExitButton(this.farmWorld),1008,16);

        // menambahkan objek untuk menampilkan koin
        addObject(new MoneyIcon(),750,120);
        addObject(new MoneyText(),850,120);

        //
        Data data = new Data();
        for (int i = 0; i < data.items.length; i++) {
            if (i == 0){
                this.item = new Radish();
            }
            else if(i == 1){
                this.item = new Carrot();
            }
            else if(i == 2){
                this.item = new RadishSeed();
            }
            else if(i == 3){
                this.item = new CarrotSeed();
            }
            // use Placeholder if no item is assigned to inventory slot
            else {
                this.item = new Placeholder();
            }
            // Menyesuaikan nilai dasar/posisi pada sumbu y dari item saat pindah baris (ketika mencapai ID 8 atau 16
            if (i % 8 == 0 && i != 0) {
                this.baseYImg += shift;
                this.baseYName += shift;
                this.baseYPrice += shift;
                // "reset" shift on new row
                this.iMod += 8;
            }

            // Membuat objek teks dengan data dari file save-an.
            this.textName = new Text(this.item.getName(), 15);
            this.textPrice = new Text("Cost: "+this.item.getBuyPrice(), 15);

            // Menggeser sumbu x.
            int xAdd = (i-this.iMod)*this.shift;

            addObject(this.item, this.baseX+xAdd, this.baseYImg);
            addObject(this.textName, this.baseX+xAdd, this.baseYName);
            addObject(this.textPrice, this.baseX+xAdd, this.baseYPrice);
        }
    }
}
```

Pada kelas “ShopUi”, yang merupakan subclass dari kelas Greenfoot "World". Tujuannya adalah untuk menampilkan antarmuka pengguna (UI) untuk toko dalam dunia permainan. kelas “ShopUi” bertanggung jawab untuk menampilkan toko dalam dunia permainan, menunjukkan item-item yang tersedia untuk dibeli, menampilkan teks informasi tentang setiap item, dan memungkinkan interaksi terkait pembelian kepada pemain. Ini memberikan pengalaman berbelanja dalam permainan, di mana pemain dapat melihat item yang tersedia dan membeli item sesuai dengan kebutuhan mereka.

6. Berikut merupakan tampilan code dari kelas abstrak bernama “Animal”

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

public abstract class Animal extends Actor
{

}
```

Dalam implementasi yang lebih lanjut, kelas “Animal” dapat digunakan sebagai induk bagi berbagai jenis hewan, pada game Green Field kami menggunakan kelasi ini sebagai induk dari kelas “horse”.

7. Berikut merupakan tampilan code dari kelas bernama “Horse”

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

public class Horse extends Animal
{
    // Menambahkan suara untuk makan wortel
    private GreenfootSound carrotSound = new GreenfootSound("sound1.mp3");

    // Menambahkan suara untuk makan lobak
    private GreenfootSound radishSound = new GreenfootSound("sound2.mp3");

    public void act()
    {
        if(Greenfoot.mouseClicked(this)) {
            // memberikan makan ke kuda
            this.feedHorse();
        }
    }

    // coding yang di eksekusi saat memberi makan kuda
    private void feedHorse(){
        Data data = new Data();
        if (data.items[1] > 0){
            // hilangkan 1 wortel saat memberi makan kuda
            data.items[1] -= 1;
            data.write();

            Carrot carrot = new Carrot();
            getWorld().addObject(carrot, 25, 2);
```

```
            Carrot carrot = new Carrot();
            getWorld().addObject(carrot, 25, 2);

            Greenfoot.delay(80);
            getWorld().removeObject(carrot);

            // animasi saat kuda dikasilh makan
            this.turn(20);
            Greenfoot.delay(20);
            this.turn(-20);

            // Memutarkan suara wortel
            carrotSound.play();
        }
        else if (data.items[2] > 0) {
            // hilangkan 1 lobak saat memberi makan kuda
            data.items[2] -= 1;
            data.write();

            Radish radish = new Radish();
            getWorld().addObject(radish, 25, 2);

            Greenfoot.delay(80);
            getWorld().removeObject(radish);

            // animasi saat memberi makan kuda
            this.turn(20);
            Greenfoot.delay(20);
            this.turn(-20);

            // Memutarkan suara lobak
            radishSound.play();
        }
    }
}
```

Class compiled - no syntax errors saved

Potongan kode diatas adalah kelas “Horse” yang merupakan turunan dari kelas “Animal” dalam lingkungan Greenfoot. kelas “Horse” diimplementasikan untuk merespons klik mouse, memeriksa dan mengurangi jumlah wortel atau lobak dari inventaris, menampilkan efek visual makanan yang diberikan kepada kuda, melakukan animasi rotasi kuda, dan memutar suara sesuai dengan jenis makanan yang diberikan. Ini adalah bagian dari interaksi antara pemain dan karakter kuda dalam dunia permainan Greenfoot.

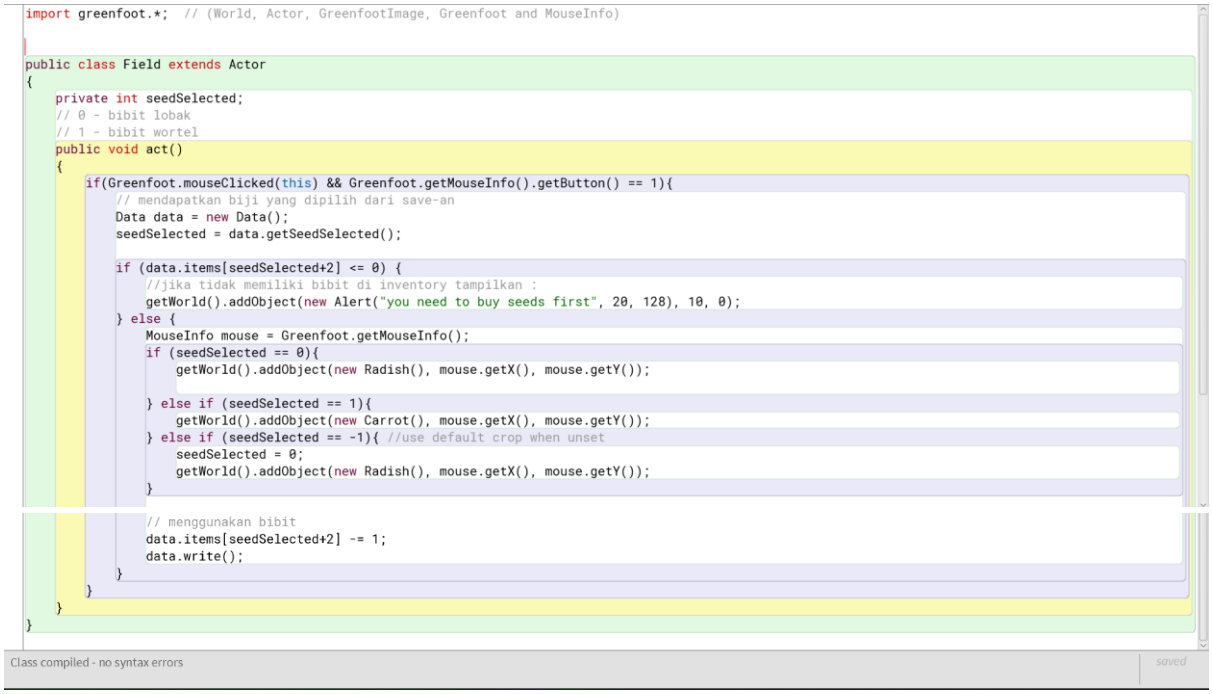
8. Berikut merupakan tampilan code dari kelas bernama “Data”

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
import java.io.File;  // mengimpor file kelas
import java.io.FileWriter;
import java.io.IOException;

import java.io.FileNotFoundException;  // mengimport kelas ini untuk menghandle error
import java.util.Scanner;

/**
 * Data class holds some game data.
 */
public class Data extends Actor
{
    String filename = "save.txt";

    public static int[] items = new int[24];
    // id inventory/item
    // 0 - lobak
    // 1 - wortel
    // 2 - bibit lobak
    // 3 - bibit wortel

    // uang, pemilihan bibit
    private static int[] data = {0,-1};

    // membuat save file apabila tidak ada data save yang di load atau di read
```

```
public Data(){
    try {
        File file = new File(this.filename);
        //membaca file apabila ada
        if (!file.createNewFile()) {
            this.read();
        }
    } catch(IOException e){}
    this.write();
}

//menulis data file, dengan nilai return true apa bila berhasil
public boolean write(){
    try {
        FileWriter writer = new FileWriter(this.filename);
        writer.write(Integer.toString(this.data[0]) + "\n");
        writer.write(Integer.toString(this.data[1]) + "\n");
        writer.write(this.arrayToString(items));
        writer.close();
        return true;
    } catch (IOException e) {return false;}
}

//membaca data
public void read(){
    try {
        File file = new File(this.filename);
        Scanner reader = new Scanner(file);
        int i = 0;
        while (i < 3) {
            String tempdata = reader.nextLine();
            if(i<2){
                this.data[i] = Integer.parseInt(tempdata);
            }
            else if (i==2){
                this.items = this.stringToArray(tempdata);
            }
            i += 1;
        }
        reader.close();
    } catch (FileNotFoundException e) {}
}

public int getMoney(){return this.data[0];}

public void setMoney(int amount){
    this.data[0] = amount;
    this.write();
}

public void addMoney(int amount){
    this.data[0] += amount;
    this.write();
}

public int getSeedSelected(){return this.data[1];}

public void setSeedSelected(int seed){
    this.data[1] = seed;
    this.write();
}

public void addItem(int id, int count) {
    this.items[id] += count;
    this.write();
}

private String arrayToString(int[] array){
    String string = "";
    // Mengulangi array dan mengembalikan string yang dipisahkan oleh koma
    for (int i = 0; i < array.length; i++) {
        string += Integer.toString(array[i])+",";
    }
    return string;
}

private int[] stringToArray(String string){
    // Memisahkan string berdasarkan koma (,), dan menaruh nilainya ke dalam stringArr.
    String[] stringArr = string.split(",");
    // Mengulangi array integer yang baru dan menyimpan string sebagai bilangan bulat dalam intArr.
    int[] intArr = new int[24];
    for (int i = 0; i < intArr.length; i++) {
        intArr[i] = Integer.parseInt(stringArr[i]);
    }
    return intArr;
}
float i = 0.5f;
```

Pada kelas “Data” yang bertanggung jawab untuk menyimpan dan memanipulasi data permainan, seperti inventaris, uang, dan status lainnya yang diperlukan dalam permainan. Kelas “Data” digunakan untuk mengelola data penting yang digunakan dalam permainan. Ini membantu dalam menyimpan dan memulihkan status permainan saat permainan dimulai ulang atau di-load kembali.

9. Berikut merupakan tampilan code dari kelas bernama “Field”



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

public class Field extends Actor
{
    private int seedSelected;
    // 0 - bibit lobak
    // 1 - bibit wortel
    public void act()
    {
        if(Greenfoot.mouseClicked(this) && Greenfoot.getMouseInfo().getButton() == 1){
            // mendapatkan biji yang dipilih dari save-an
            Data data = new Data();
            seedSelected = data.getSeedSelected();

            if (data.items[seedSelected+2] <= 0) {
                //jika tidak memiliki bibit di inventory tampilkan :
                getWorld().addObject(new Alert("you need to buy seeds first", 20, 128), 10, 0);
            } else {
                MouseInfo mouse = Greenfoot.getMouseInfo();
                if (seedSelected == 0){
                    getWorld().addObject(new Radish(), mouse.getX(), mouse.getY());

                } else if (seedSelected == 1){
                    getWorld().addObject(new Carrot(), mouse.getX(), mouse.getY());
                } else if (seedSelected == -1){ //use default crop when unset
                    seedSelected = 0;
                    getWorld().addObject(new Radish(), mouse.getX(), mouse.getY());
                }

                // menggunakan bibit
                data.items[seedSelected+2] -= 1;
                data.write();
            }
        }
    }
}
```

Class compiled - no syntax errors saved

Pada Kelas “Field” digunakan untuk menangani logika penanaman bibit di lahan. Ini memungkinkan pemain untuk menanam bibit di lokasi yang mereka pilih dalam dunia permainan, dengan mempertimbangkan jenis bibit yang dipilih dan ketersediaan bibit dalam inventaris.

10. Berikut merupakan tampilan code dari kelas bernama “Interface”

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

public abstract class Interface extends Actor
{
}
```

Kode diatas adalah deklarasi kelas abstrak “Interface”. Kelas abstrak seperti ini dapat digunakan sebagai landasan untuk membuat antarmuka pengguna atau komponen interaktif lainnya dalam permainan. Kelas turunannya akan menambahkan perilaku dan fungsi yang diperlukan untuk interaksi dengan pemain atau elemen-elemen lain dalam dunia permainan.

11. Berikut merupakan tampilan code dari kelas bernama “Alert”

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

public class Alert extends Interface
{
    private int counter;
    private int time;

    private String text = "";
    private int size = 10;
    public Alert(String text, int size, int time){
        this.text = text;
        this.size = size;
        this.time = time;
    }

    public void act()
    {
        // Setel teks peringatan dan hapus setelah jangka waktu tertentu.
        setImage(new GreenfootImage(this.text, this.size, Color.WHITE, new Color(0,0,0,0)));
        if (counter < time){
            counter++;
        } else {
            getWorld().removeObject(this);
        }

    }
}
```

Kode diatas adalah kelas “Alert” yang merupakan turunan dari kelas “Interface”, kelas “Alert” ini dapat digunakan untuk menampilkan pesan peringatan yang sementara di dunia permainan. Pesan ini mungkin memberi petunjuk, informasi, atau peringatan kepada pemain dalam konteks permainan.

12. Berikut merupakan tampilan code dari kelas bernama “ExitButton”

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

public class ExitButton extends Interface
{
    World exitTo;
    public ExitButton(World world) {
        this.exitTo = world;
    }

    public void act()
    {
        if(Greenfoot.mouseClicked(this) || Greenfoot.isKeyDown("escape")) {
            // keluar dari world konstruktor saat berada di toko inventory atau help.
            if (getWorld() instanceof InventoryUi || getWorld() instanceof ShopUi || getWorld() instanceof Help) {
                Greenfoot.setWorld(exitTo);
            }
        }
    }
}
```

Kode yang ditunjukkan adalah kelas “ExitButton”, yang merupakan turunan dari kelas “Interface”. digunakan sebagai kontrol untuk keluar dari suatu dunia dalam permainan, khususnya saat berada di dunia InventoryUi, ShopUi, atau Help. Ini memberikan fungsionalitas bagi pemain untuk kembali ke dunia sebelumnya dengan menekan tombol keluar atau tombol "escape" saat berada di dunia tertentu dalam permainan.

13. Berikut merupakan tampilan code dari kelas bernama “InventoryButton”

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

/**
 * Button to change world to Inventory.
 */
public class InventoryButton extends Interface
{
    public void act()
    {
        if(Greenfoot.mouseClicked(this)) {
            //mengubah world ke world inventory dan menyediakan world saat ini sebagai argumen untuk dikembalikan nanti
            Greenfoot.setWorld(new InventoryUi((Farm)this.getWorld()));
        }

        // Mengubah gambar tombol saat disorot
        if (Greenfoot.mouseMoved(this)){
            setImage(new GreenfootImage("backpack-outline.png"));
        }
        if (Greenfoot.mouseMoved(null) && !Greenfoot.mouseMoved(this)){
            setImage(new GreenfootImage("backpack.png"));
        }
    }
}
```

Kode diatas adalah kelas “InventoryButton”, yang merupakan turunan dari kelas “Interface”. Kelas “InventoryButton” digunakan sebagai kontrol yang memungkinkan pemain untuk beralih ke dunia inventaris dalam permainan. Selain mengatur transisi dunia, kelas ini juga memberikan respons visual dengan mengubah gambar tombol saat disorot oleh mouse, memberikan umpan balik visual yang berguna bagi pemain.

14. Berikut merupakan tampilan code dari kelas bernama “MoneyIcon”

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
public class MoneyIcon extends Interface
{
    int counter = 0;
    int image = 0;
    GreenfootImage[] images = {
            new GreenfootImage("coin_01.png"),
            new GreenfootImage("coin_02.png"),
            new GreenfootImage("coin_03.png"),
            new GreenfootImage("coin_04.png"),
            new GreenfootImage("coin_05.png"),
            new GreenfootImage("coin_06.png"),
            new GreenfootImage("coin_07.png"),
            new GreenfootImage("coin_08.png")
        };
    boolean animation = false;
    public void act()
    {
        // Jalankan animasi saat diaktifkan
        if (animation != false) {
            if(this.counter % 10 == 0) {
                this.setImage(this.images[this.image]);
                this.image = this.image+1;
                if (this.image == this.images.length) {
                    this.image = 0;
                }
            }
            this.counter = this.counter+1;
        }
    }
```

Kode diatas dalah kelas “MoneyIcon”, turunan dari kelas “Interface”.

15. Berikut merupakan tampilan code dari kelas bernama “MoneyText”

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

public class MoneyText extends Interface
{
    public void act()
    {
        // Mengambil uang dari save dan menampilkannya
        Data data = new Data();
        setImage(new GreenfootImage("" + data.getMoney(), 40, Color.WHITE, new Color(0,0,0,0)));

        if(Greenfoot.mouseClicked(this)) {
            data.addMoney(100);
        }
    }
}
```

Kode diatas merupakan Kelas “MoneyText” adalah turunan dari kelas Interface. Kelas **MoneyText** bertujuan untuk menampilkan jumlah uang yang ada dalam permainan sebagai teks pada layar. Selain menampilkan jumlah uang,

16. Berikut adalah tampilan code dari kelas “SeedSelectButton”

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

/**
 * Button used for changing the currently selected seed for planting.
 */
public class SeedSelectButton extends Interface
{
    Data data = new Data();

    Text seedSelectText;
    boolean[] seeds = {true, false};
    // 0 - lobak
    // 1 - wortel

    int selected = 0;

    public SeedSelectButton(Text text) {
        this.seedSelectText = text;

        // Dapatkan biji yang saat ini dipilih dari save ketika bukan pada save baru
        if(this.data.getSeedSelected() != -1){
            for (int i = 0; i < this.seeds.length; i++) {
                if (i==this.data.getSeedSelected()) {
                    this.seeds[i] = true;
                    this.selected = i;
                } else{
                    this.seeds[i] = false;
                }
            }
            this.updateText();
        } else{
            //
            this.data.setSeedSelected(0);
            this.selected = 0;
        }
    }

    public void act()
    {
        if(Greenfoot.mouseClicked(this)){
            // Mendapatkan biji yang dipilih dan mengatur semua nilai menjadi salah
            for (int i = 0; i < this.seeds.length; i++) {
                if(this.seeds[i] == true) {
                    selected = i;
                    this.seeds[i] = false;
                    break;
                }
            }

            // Mengatur biji berikutnya atau pertama (jika berada di akhir) yang tersedia menjadi yang dipilih
            try {
                this.selected += 1;
                this.seeds[selected] = true;
            }
            catch(ArrayIndexOutOfBoundsException e) {
                this.seeds[0] = true;
                this.selected = 0;
            }

            this.updateText();
            this.data.setSeedSelected(selected);
        }

        private void updateText() {
            if(this.seeds[0]) {
                this.seedSelectText.setText("Radish");
            }
            else if(this.seeds[1]) {
                this.seedSelectText.setText("Carrot");
            }
        }
    }
}
```

Kode diatas digunakan untuk menggunakan tombol pemilihan biji di game ini. Kelas ini mengatur pemilihan biji, baik lobak maupun wortel, dan memperbarui tampilan teks. Dengan mengklik tombol mouse dapat mengubah pilihan biji dan menyimpan.

17. Berikut adalah tampilan dari kelas “ShopButton”

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

/**
 * Button to change world to Shop.
 */
public class ShopButton extends Interface
{
    public void act()
    {
        if(Greenfoot.mouseClicked(this)) {
            //mengubah world ke world shop dan menyediakan world saat ini sebagai argumen untuk dikembalikan nanti
            Greenfoot.setWorld(new ShopUi((Farm)this.getWorld()));
        }
    }
}
```

Kode ini menjalankan tombol yang digunakan dalam game untuk mengakses tampilan toko. Ketika tombol ini diklik, World permainan berubah menjadi World toko, atau “ShopUi”. Dengan fitur ini, pemain dapat mengakses toko dari dalam permainan, yang menawarkan nuansa interaktif dan kemudahan navigasi.

18. Berikut adalah tampilan dari kelas “Text”

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

/**
 * Ui class to simplify displaying text.
 */
public class Text extends Interface
{
    private String text = "";
    private int size = 10;
    public Text(String text, int size){
        this.text = text;
        this.size = size;

    }

    public void act(){
        // Setel gambar menjadi teks dengan parameter yang diberikan
        setImage(new GreenfootImage(this.text, this.size, Color.WHITE, new Color(0,0,0,0)));
    }

    public void setText(String text){
        this.text = text;
    }

    public void setSize(int size){
        this.size = size;
    }
}
```

Kode ini mengaktifkan kelas “Text”, yang dimaksudkan untuk membuat interface pengguna game lebih mudah untuk melihat teks. Kelas ini menawarkan metode untuk mengatur teks dan ukuran, memberikan fleksibilitas dalam menampilkan informasi teks dengan mudah.

19. Berikut adalah tampilan dari kelas “Item”

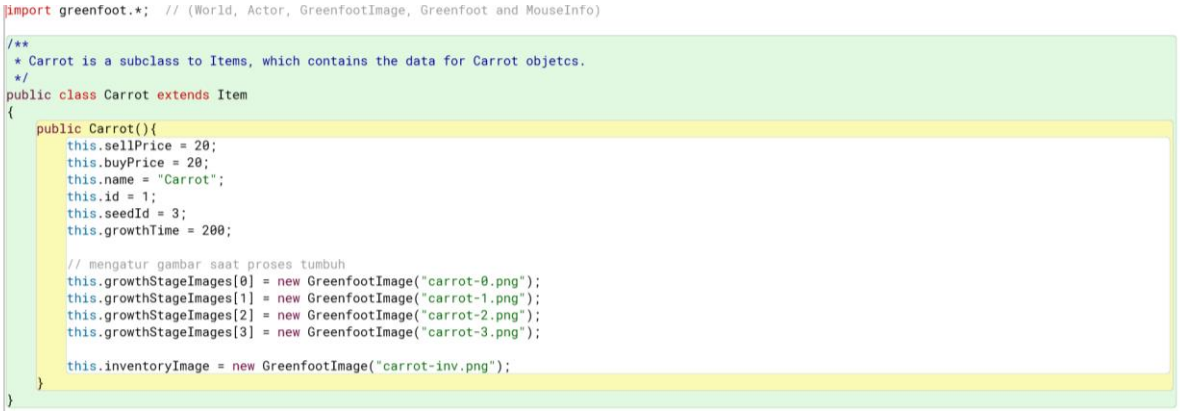
```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Items superclass holds all items used in the games as subclasses.
 */
public abstract class Item extends Actor
{
    int sellPrice;
    int buyPrice;
    String name;
    int id;
    int counter = 0;
    int growthStage = 0;
    int growthTime = 0;
    int seedId;

    GreenfootImage[] growthStageImages = new GreenfootImage[4];
    GreenfootImage inventoryImage;
```

```java
public void act()
{
    if (getWorld() instanceof Farm) {
        this.counter += 1;
        // change growth change after certain time period
        if (counter % growthTime == 0 && growthStage < 3) {
            this.growthStage += 1;
        }
        this.setImage(this.growthStageImages[growthStage]);

        // right click to harvest crop (add to inventory, add seed to inventory, then remove from world)
        if(Greenfoot.mouseClicked(this) && Greenfoot.getMouseInfo().getButton() == 3 && this.growthStage == 3) {
            Data data = new Data();
            data.addItem(this.id, 1);
            data.addItem(this.seedId, 1);
            getWorld().removeObject(this);
        }
    }
    // set image when in inventory or shop
    if (getWorld() instanceof InventoryUi) {
        this.setImage(this.inventoryImage);
        // add sell option when in inventory
        if (Greenfoot.mouseClicked(this)){this.sell();}
    }
    if (getWorld() instanceof ShopUi) {
        this.setImage(this.inventoryImage);
        // add buy option when in shop
        if (Greenfoot.mouseClicked(this)){this.buy();}
    }
}
```

```java
int getSellPrice() {return this.sellPrice;}

int getBuyPrice() {return this.buyPrice;}

// sell action for items in inventory
void sell(){
    Data data = new Data();
    if (data.items[this.id] > 0){
        // add money, remove item from inventory and save to file
        data.addMoney(this.sellPrice);
        data.items[this.id] -= 1;
        data.write();
    }
    // refresh inventory after selling
    Farm farmWorld = ((InventoryUi)getWorld()).farmWorld;
    Greenfoot.setWorld(new InventoryUi(farmWorld));
}

// buy action for items in shop
void buy(){
    // add item to inventory, remove money and save to file
    Data data = new Data();
    if(data.getMoney() >= this.buyPrice) {
        data.items[this.id] += 1;
        data.setMoney(data.getMoney()-this.buyPrice);
        data.write();
    }
}
}
```

Kelas “Item” adalah kelas abstrak yang berfungsi sebagai dasar untuk semua item dalam permainan. Logika pertumbuhan, interaksi jual beli, dan representasi visual item diWorld permainan, inventory, dan toko adalah semua bagian dari ini.

20. Berikut adalah tampilan dari kelas “Carrot”



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

/**
 * Carrot is a subclass to Items, which contains the data for Carrot objetcs.
 */
public class Carrot extends Item
{
    public Carrot(){
        this.sellPrice = 20;
        this.buyPrice = 20;
        this.name = "Carrot";
        this.id = 1;
        this.seedId = 3;
        this.growthTime = 200;

        // mengatur gambar saat proses tumbuh
        this.growthStageImages[0] = new GreenfootImage("carrot-0.png");
        this.growthStageImages[1] = new GreenfootImage("carrot-1.png");
        this.growthStageImages[2] = new GreenfootImage("carrot-2.png");
        this.growthStageImages[3] = new GreenfootImage("carrot-3.png");

        this.inventoryImage = new GreenfootImage("carrot-inv.png");
    }
}
```

Subkelas Item, Kelas Wortel menyimpan data khusus untuk objek wortel yang ada dalam permainan. Ini mencakup pengaktifan fitur seperti harga jual, harga beli, nama, identitas, waktu pertumbuhan, dan gambar yang menunjukkan tahap pertumbuhan dan inventaris. Kelas ini dapat digunakan untuk membuat objek wortel dengan karakteristik tertentu dengan menggunakan hierarki pewarisan.

21. Berikut adalah tampilan dari kelas “CarrotSeed”

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

/**
 * CarrotSeed is a subclass to Items, which contains the data for CarrotSeed objetcs.
 */
public class CarrotSeed extends Item
{
    public CarrotSeed(){
        this.sellPrice = 0;
        this.buyPrice = 10;
        this.name = "Carrot\nSeed";
        this.id = 3;
        this.inventoryImage = new GreenfootImage("carrot-0.png");
    }

    @Override
    public void act(){
        // mengatur gambar saat di inventory dan di shop
        if (getWorld() instanceof InventoryUi) {
            this.setImage(this.inventoryImage);
            // menambahkan opsi penjualan saat di invetory
            if (Greenfoot.mouseClicked(this)){this.sell();}
        }
        if (getWorld() instanceof ShopUi) {
            this.setImage(this.inventoryImage);
            // menambahkan opsi beli saat di shop
            if (Greenfoot.mouseClicked(this)){this.buy();}
        }
    }
}
```

Kelas “CarrotSeed” merupakan subkelas dari Item dan menyimpan data spesifik untuk objek biji wortel dalam permainan. Kelas ini dapat mengontrol bagaimana gambar objek biji wortel ditampilkan saat berada di dalam toko atau dalam inventaris dengan menggunakan metode act(). Untuk membagi fungsionalitas umum yang dimiliki oleh semua jenis item dalam permainan, kelas ini, seperti kelas “Carrot”, juga menggunakan hierarki pewarisan.

22. Berikut adalah tampilan dari kelas “PlaceHolder”

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

/**
 * Placeholder acts as a placeholder for not existing items in inventory and shop.
 */
public class Placeholder extends Item
{
    public Placeholder(){
        this.sellPrice = 0;
        this.name = "N/A";
    }
}
```

Kelas “Placeholder” dibuat untuk digunakan sebagai representasi item yang tidak ada dalam inventory atau toko. Item placeholder memiliki harga jual yang diatur ke 0 dan namanya diatur sebagai "N/A", yang membantu mengelola situasi di mana item mungkin tidak tersedia atau belum dibuat dalam konteks permainan.

23. Berikut adalah tampilan dari kelas “Radish”

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

/**
 * Radish is a subclass to Items, which contains the data for Radish objetcs.
 */
public class Radish extends Item
{
    public Radish(){
        this.sellPrice = 20;
        this.buyPrice = 20;
        this.name = "Radish";
        this.id = 0;
        this.seedId = 2;
        this.growthTime = 100;

        // mengatur gambar saat proses tumbuh
        this.growthStageImages[0] = new GreenfootImage("radish-0.png");
        this.growthStageImages[1] = new GreenfootImage("radish-1.png");
        this.growthStageImages[2] = new GreenfootImage("radish-2.png");
        this.growthStageImages[3] = new GreenfootImage("radish-3.png");

        this.inventoryImage = new GreenfootImage("radish-inv.png");
    }
}
```

Kelas “Radish” merupakan subkelas dari Item dan menyimpan data spesifik untuk objek lobak dalam permainan. Ini mencakup pengaktifan fitur seperti harga jual, harga beli, nama, identitas, waktu pertumbuhan, dan gambar yang menunjukkan tahap pertumbuhan dan inventory. Untuk membagi fungsionalitas umum yang dimiliki oleh semua jenis item dalam permainan, kelas ini juga menggunakan hierarki pewarisan, seperti yang dilakukan oleh kelas “Carrot”.

24. Berikut adalah tampilan dari kelas “RadishSeed”

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

/**
 * RadishSeed is a subclass to Items, which contains the data for RadishSeed objetcs.
 */
public class RadishSeed extends Item
{
    public RadishSeed(){
        this.sellPrice = 0;
        this.buyPrice = 10;
        this.name = "Radish\nSeed";
        this.id = 2;
        this.inventoryImage = new GreenfootImage("radish-0.png");
    }

    @Override
    public void act(){
        // mengatur gambar saat di inventory dan di shop
        if (getWorld() instanceof InventoryUi) {
            this.setImage(this.inventoryImage);
            // menambahkan opsi penjualan saat di invetory
            if (Greenfoot.mouseClicked(this)){this.sell();}
        }
        if (getWorld() instanceof ShopUi) {
            this.setImage(this.inventoryImage);
            // menambahkan opsi beli saat di shop
            if (Greenfoot.mouseClicked(this)){this.buy();}
        }
    }
}
```

Kelas “RadishSeed” merupakan subkelas dari Item dan menyimpan data spesifik untuk objek biji lobak dalam permainan. Kelas ini dapat mengontrol bagaimana gambar objek biji Radish ditampilkan ketika berada di dalam toko atau dalam inventaris dengan menggunakan metode act(). Selain itu, kelas ini menggunakan hierarki pewarisan untuk membagi fungsionalitas umum yang dimiliki oleh semua jenis item dalam permainan, seperti yang dilakukan oleh kelas “CarrotSeed”.

Kesimpulan

Pemrograman Berorientasi Objek (PBO) adalah model pemrograman yang mengutamakan pengorganisasian dan struktur kode yang didasarkan pada konsep objek. Program dibangun sebagai kumpulan objek yang saling berinteraksi. Pembuatan kelas dan objek adalah komponen utama PBO. Kelas adalah blueprint atau cetak biru yang mendefinisikan sifat dan perilaku.

PBO sering digunakan dalam pengembangan game untuk memodelkan objek-objek seperti karakter, item, dan lingkungan permainan. Penggunaan PBO mempermudah pengorganisasian kode game, meningkatkan keterbacaan, dan menyederhanakan proses pengembangan.