LAPORAN TUGAS BESAR
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK SEMESTER GANJIL 2020-2021

Disusun oleh:

Ivan Andrianto (F1D018027)

Majidi Aprizan (F1D018035)

Farah Tria Ningrum (F1D018073)

Asisten Pembimbing:

Rizaldi Septian Fauzi (F1D017075)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MATARAM

2020

LEMBAR PENGESAHAN

LAPORAN TUGAS BESAR

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Dikerjakan oleh:

Ivan Andrianto (F1D018027)

Majidi Aprizan (F1D018035)

Farah Tria Ningrum (F1D018073)

Mengetahui

Dosen Pengampu, **Koordinator Asisten,**

(Royana Afwani, S.T., M.T.) **(I Putu Pranata Kusuma Yuda)**

NIP: 19850707 201404 2 001 **NIM: F1D017032**

Kepala Laboratorium Praktikum,

(Nadiyasari Agitha, S.Kom., M.M.T.)
NIP: 19860813 201803 2 001

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Beberapa bulan lalu dunia digemparkan dengan berita tentang menyebarnya suatu wabah virus yang berasal dari Wuhan, China. Virus ini bernama SARS-CoV-2, pada tahun 2002 virus yang hampir mirip pernah muncul di China tetapi penyebarannya cepat dibendung, bernama SARS-CoV. Hanya sekitar 3000-an kasus penyebaran wabah SARS hingga tahun 2016 yang pernah tercatat dan tidak pernah mengalami peningkatan. Singkatnya virus tersebut berevolusi selama belasan tahun terakhir dan muncul kembali dengan kelelawar sebagai inang atau *carrier*-nya. Penyebaran juga didorong oleh tertanamnya budaya serta kebiasaan masyarakat China dalam mengonsumsi berbagai hewan yang tidak biasa seperti: kucing, tikus, ular, buaya, dan lain sebagainya. Hal ini menyebabkan penyebaran virus COVID-19 meningkat sangat signifikan. Tercatat terdapat 82.871 kasus penularan COVID-19 di Cina dengan jumlah pasien sembuh sebanyak 77.550 orang dan telah memakan korban jiwa sebanyak 4.900 jiwa [1].

Di Indonesia, jumlah kasus penyebaran COVID-19 yang tercatat sangatlah banyak. Mengutip data Worldometers, selasa (20/10/2020) kasus terkonfirmasi positif virus corona di Indonesia bertambah sebanyak 58.120 kasus selama dua minggu terakhir, terhitung sejak tanggal 7 Oktober 2020. Dengan begitu, total jumlah kasus aktif virus corona tercatat sejak awal bulan April ada sebanyak 373 ribu kasus. Jumlah ini menempatkan Indonesia sebagai negara ke-2 dengan jumlah kasus COVID-19 terbanyak se-Asia Tenggara, setelah Vietnam [2]. Hal ini disebabkan oleh ketidakpedulian masyarakat akan himbauan pemerintah tentang bahaya penyebaran wabah COVID-19 dan berbagai protokol pencegahan penularan yang telah dianjurkan oleh pemerintah, sehingga masyarakat dengan tanpa sadar menjadi perantara penyebaran wabah COVID 19.

Oleh karena itu kami berniat untuk membuat sebuah *game* sederhana yang dapat meningkatkan kesadaran masyarakat terhadap bahaya dari virus COVID-19. *Game* yang akan dibuat dan dikembangkan berjudul “Whack Corona”, bergenre-kan *casual-informative game,* dengan cara main dan UI yang sederhana namun tetap memerlukan penyelesaian yang kompleks dan menantang di setiap levelnya, serta ditambah dengan penerapan tema COVID-19 sehingga diharapkan *game* “Whack Corona” tidak hanya memberikan hiburan semata namun juga

mengedukasi para *player* agar semakin *aware* terhadap wabah COVID-19. *Game* ini dikembangkan menggunakan bahasa pemrograman Java yang sudah mendukung *Object Oriented Programming* (OOP).

1.2 Deskripsi Aplikasi

Permainan "Whack Corona" merupakan permainan yang terdiri dari satu orang pemain atau *single player*. Permainan ini terinspirasi dari permainan "Whack-a-mole" yang dikembangkan sesuai dengan keadaan pandemi saat ini. Permainan ini memiliki 4 level dengan durasi permainan selama 15 detik tiap levelnya. Semakin tinggi level, maka tingkat kesulitan pun akan semakin tinggi. Pemain harus mengumpulkan skor hingga jumlah tertentu agar bisa melanjutkan ke level selanjutnya. Cara bermainnya sangat mudah yaitu: dengan membasmi virus yang bermunculan untuk meraih skor. Tantangan permainan ini terletak pada virusnya yang bervariasi dengan *reward* dan *punishment* yang berbeda-beda. Prosedur dari permainan yang akan dibuat dapat diuraikan sebagai berikut:

1. Program akan menampilkan menu utama dari permainan "Whack Corona" berupa 3 menu penting yaitu: menu *Start*, menu *High Score*, dan menu *Quit*.
2. Jika pemain memilih menu *Play*, maka akan muncul *window* baru yang menampilkan area permainan beserta rincian waktu, skor, level, dan keterangan dari virus yang dibasmi. Kemudian juga muncul pesan mengenai info permainan pada level tersebut. Begitu pemain menekan tombol OK, maka durasi akan hitung mundur sebagai penanda bahwa permainan telah dimulai dan pemain harus mengumpulkan skor sebanyak-banyaknya dengan membasmi virus yang muncul. Ketika durasi telah berakhir maka akan muncul pesan mengenai keterangan apakah skor yang dicapai telah memenuhi target untuk lanjut atau tidak.
3. Ketika permainan telah berakhir, baik itu karena permainan telah diselesaikan atau *Game over*, maka akan muncul *window* baru yang meminta pemain untuk memasukkan nama. Nama yang dimasukkan beserta akumulasi skor yang telah dimainkan pada tiap level akan disimpan ke dalam *database*. Sepuluh skor terbaik yang telah tersimpan dalam *database* akan ditampilkan pada menu *High Score*.

4. Ketika nama telah dimasukkan, maka akan muncul *window* yang menanyakan apakah pemain ingin bermain lagi atau tidak. Jika iya, maka permainan akan dimulai kembali dari level 1. Jika tidak, maka akan keluar dari permainan.
5. Jika pemain memilih menu *High Score*, maka akan muncul *window* berupa 10 informasi nama pemain dengan skor tertinggi yang telah dicapai pada permainan tersebut.
6. Jika pemain memilih menu *Quit*, maka program akan berhenti.

1.3 Batasan Masalah

Terdapat batasan-batasan dalam pembuatan program "Whack Corona", di antaranya yaitu :

1. Pemain dapat mengatur tombol yang digunakan sebagai navigasi.
2. Menyimpan dan menampilkan data dari skor tertinggi pemain pada *database*.
3. Terdapat penerapan *inheritance* serta implementasi konsep OOP lainnya pada kode program**.**

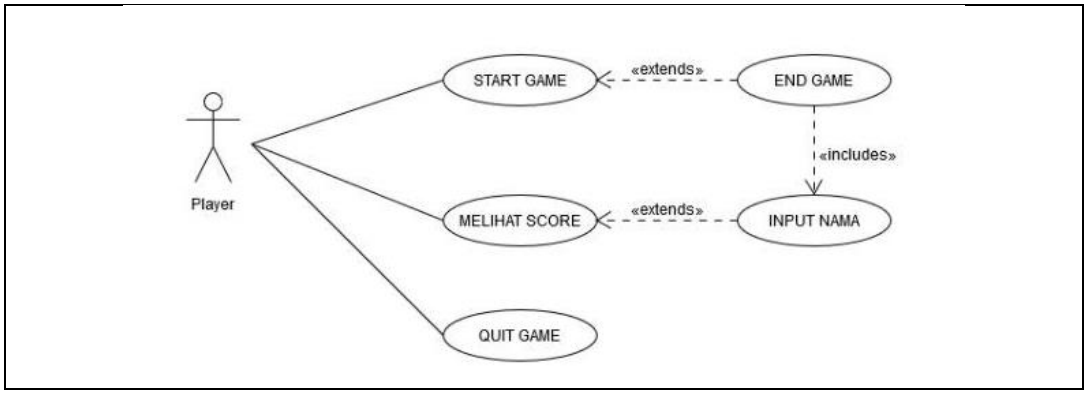
1.4 Tujuan

Terdapat beberapa tujuan dari pembuatan aplikasi "Whack Corona" yaitu sebagai berikut :

1. Sarana untuk hiburan bagi setiap kalangan.
2. Sebagai penyegar pikiran dari kepenatan akibat dari padatnya aktivitas sehari-hari.

BAB II ANALISA DAN DESAIN

2.1 Use Case Diagram

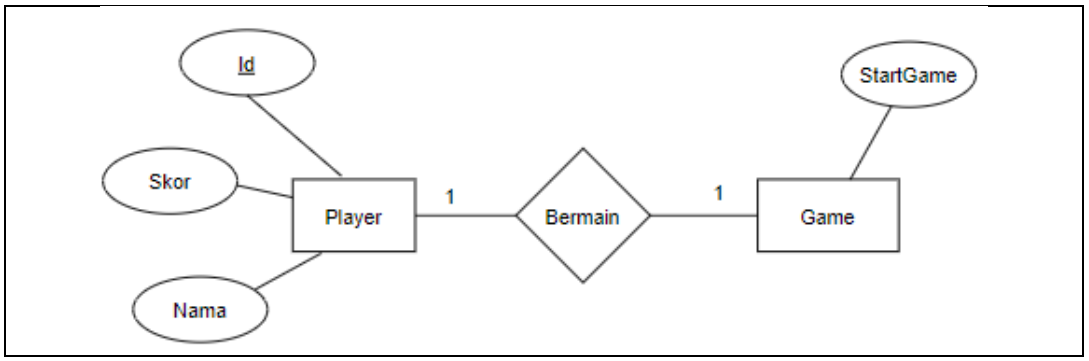


Gambar 1. *Use Case Diagram*

Berdasarkan **Gambar 1** dapat dijelaskan bahwa *Player* dapat melakukan beberapa hal, yaitu:

1. *Start Game*: Pada saat *player* memilih menu *Start Game* (*Play*), maka *player* akan mulai bermain hingga selesai. Setelah *Game Over* atau *game*-nya telah berakhir maka *player* melakukan *input* nama dan kemudian akan muncul *window* yang berisi total skor yang diraih oleh *player*.
2. Melihat *Score*: Pada saat *player* memilih menu melihat *score*, maka sistem akan menampilkan 10 skor terbaik yang telah diraih.
3. *Quit Game*: Pada saat *player* memilih menu *Quit Game*, maka *player* akan keluar dari program "Whack Corona" tersebut.

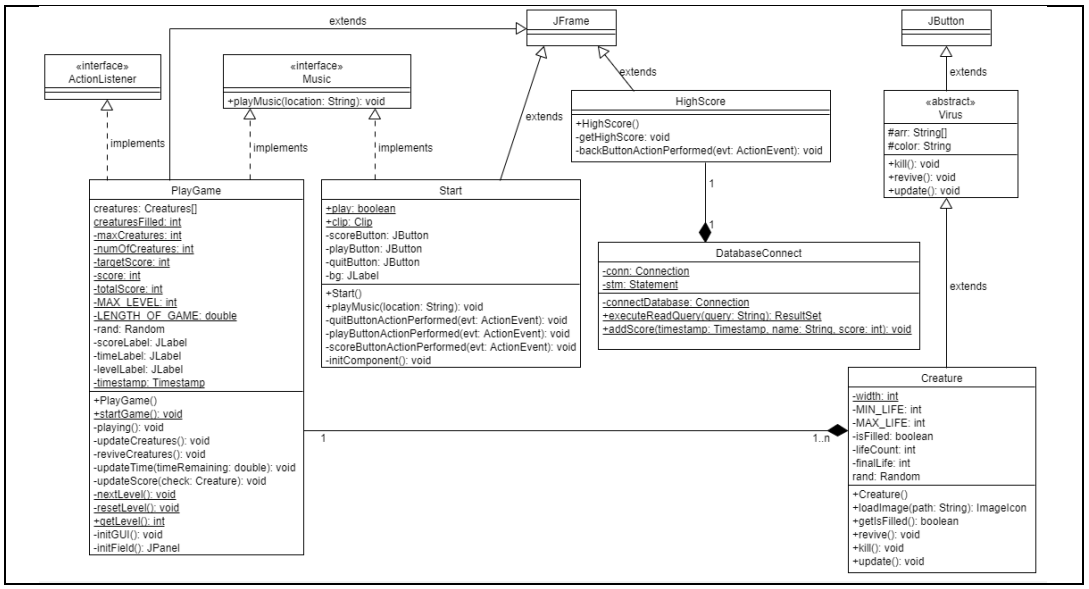
2.2 Entity Relationship Diagram (ERD)



Gambar 2. *Entity Relationship Diagram* (ERD)

Pada **Gambar 2**, merupakan *Entity Relationship Diagram* (ERD) dari program "Whack Corona". Pada ERD tersebut, terdapat 2 entitas yaitu entitas *Player* dan entitas *Game*. Pada entitas *Player*, terdapat atribut "`Nama`" , "`Skor`", dan "`Id`". Pada entitas *Game*, terdapat atribut "`StartGame`". Kardinalitas antara entitas *Player* dengan entitas *Game* yaitu 1 (*one*) ke 1 (*one*).

2.3 Class Diagram



Gambar 3. *Class Diagram*

2.4 Deskripsi Class

Gambar 3 merupakan *class* diagram yang memiliki 10 *class*. Deskripsi kelas di sini tidak mencantumkan *method* dan variabel yang otomatis dibuat oleh NetBeans IDE, seperti JFrame dan sebagainya. Berikut ini merupakan daftar dan penjelasan dari kelas-kelas tersebut beserta *method* dan variabel yang ada :

1. *Class* PlayGame

   Kelas ini digunakan untuk memainkan *game* "Whack Corona". Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

   a. PlayGame(): Merupakan *constructor* yang digunakan untuk memanggil kelas yang berisi jalannya suatu permainan.
   b. startGame(): Merupakan *method* yang digunakan untuk memainkan *game* "Whack Corona".
   c. playing(): Merupakan *method* yang berfungsi untuk mengatur hal-hal yang berhubungan dengan waktu seperti waktu dimulainya permainan, durasi permainan, *update* waktu, dan waktu yang tersisa saat permainan berlangsung.
   d. updateCreatures(): Merupakan *method* yang berfungsi untuk menambahkan virus ke dalam permainan dan virus akan bertambah jika skor memenuhi persyaratan.
   e. reviveCreatures(): *Method* yang berfungsi sebagai untuk mengatur kemunculan virus pada kotak, jika ada kotak kosong pada permainan maka virus akan muncul pada kotak tersebut.
   f. updateTime(timeRemaining: double): Berfungsi sebagai *timer* pada permainan,

pada saat waktunya mencapai di bawah 5 detik maka tulisannya akan berwarna merah.

g. updateScore(): Berfungsi untuk mengakumulasi skor pada saat bermain, jika pemain meng-klik virus berwarna hitam maka skor akan bertambah sebanyak 50, jika meng-klik berwarna merah maka skor akan berkurang sebanyak 5, jika meng-klik warna hijau maka skor akan bertambah sebanyak 10, dan jika meng-klik virus berwarna biru maka akan mengurangi poin sebanyak 30 poin.
h. nextLevel(): *Method* ini berfungsi sebagai penambah level jika pemain berhasil menempuh skor yang diatur oleh *game.* Pada saat level naik maka kotak virus akan bertambah sebanyak 3 kotak, dan target skor untuk naik ke level selanjutnya akan bertambah 50 poin.
i. resetLevel(): *Method* ini berfungsi untuk menyetel permainan ke level awal jika pemain memilih untuk main lagi saat permainan berakhir.
j. getLevel(): *Method* ini berfungsi untuk mengecek level permainan untuk menyesuaikan ukuran *frame* virus pada level tersebut.
k. initGUI(): *Method* berfungsi sebagai membangun komponen GUI dalam permainan yang berisi keterangan skor, waktu, level, deskripsi, background dan panel permainan.
l. initField(): *Method* ini berfungsi untuk mengatur panel permainan yang mana pada tiap level memiliki panel yang berbeda-beda sehingga perlu diatur pada *method* ini.
m. playMusic(location : String): *Method* ini berfungsi untuk memainkan musik pada saat kejadian tertentu

2. *Class* Start

Kelas ini berfungsi untuk menampilkan menu awal saat permainan pertama kali dibuka. Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

a. Start(): Merupakan *constructor* yang digunakan untuk memanggil kelas yang berisi jalannya suatu permainan.
b. playMusic(): *Method* ini berfungsi untuk memainkan musik saat permainan dimulai.
c. quitActionButtonPerformed(): *Method* ini berfungsi untuk keluar dari permainan.
d. playActionButtonPerformed(): *Method* ini berfungsi untuk mengalihkan pemain ke dalam permainan untuk memainkan permainan.
e. scoreActionButtonPerformed(): *Method* ini berfungsi untuk mengalihkan pemain

ke dalam menu *Highscore* untuk melihat 10 skor terbaik yang pernah dimainkan.

f. initComponent(): *Method* ini berfungsi untuk menginisialisasi komponen GUI pada tampilan menu permainan seperti *icon*, *background*, dan *button*.

3. *Class* HighScore

   Kelas ini berfungsi untuk menampilkan menu *Highscore* dalam permainan. Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

   a. HighScore(): Merupakan *constructor* yang digunakan untuk memanggil kelas yang berisi tentang segala hal yang berkaitan dengan *highscore.*
   b. getHighScore: *Method* yang berfungsi untuk memanggil data skor dalam *database* permainan untuk ditampilkan pada menu *highscore.*
   c. backButtonActionListener(): *Method* ini berfungsi sebagai tombol kembali ke menu awal permainan.

4. *Class* DatabaseConnect

   Kelas ini berfungsi untuk menyambungkan permainan dengan database. Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

   a. DatabaseConnect: *Method* ini berfungsi untuk koneksi antara permainan dan *database*.
   b. executeReadQuerry(String querry): *Method* ini berfungsi untuk membaca tabel-tabel yang digunakan dalam *database highscore.*
   c. addScore(String name,int score): *Method* ini berfungsi untuk menambahkan data skor ke dalam *database* permainan.

5. *Class* Creature

   Kelas ini berfungsi untuk menyabungkan ikon-ikon yang dibuat ke dalam permainan. Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

   a. Creature(): Merupakan *constructor* yang digunakan untuk memanggil kelas yang berisi tentang *source* permainan.
   b. loadImage(String path): *Method* ini berfungsi untuk memanggil ikon virus ke dalam permainan dengan ukuran tiap level berbeda - beda.
   c. getIsFilled(): *Method* ini berfungsi untuk memeriksa apakah kotak virus berisi atau tidak.
   d. revive(): *Method* ini berfungsi untuk mengatur kemunculan tiap-tiap virus dalam permainan serta mengacak kemunculannya pada kotak yang belum terisi.
   e. kill(): *Method* ini berfungsi untuk membunuh virus agar virus dalam kotaknya menghilang dan dapat diisi oleh virus baru lagi.

f. update(): *Method* ini berfungsi sebagai pengatur durasi lama virus muncul dalam kotak.

6. *Class* Virus

   Kelas ini berfungsi sebagai *abstract class* dalam permainan sekaligus meng-*extends* kelas JButton. Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

   a. revive(): *Method* ini berfungsi untuk mengatur kemunculan tiap-tiap virus dalam permainan serta mengacak kemunculannya pada kotak yang belum terisi.
   b. kill(): *Method* ini berfungsi untuk membunuh virus agar virus dalam kotaknya menghilang dan dapat di isi oleh virus baru lagi.
   c. update(): *Method* ini berfungsi sebagai pengatur durasi lama virus muncul dalam kotak.

7. *Class* Music

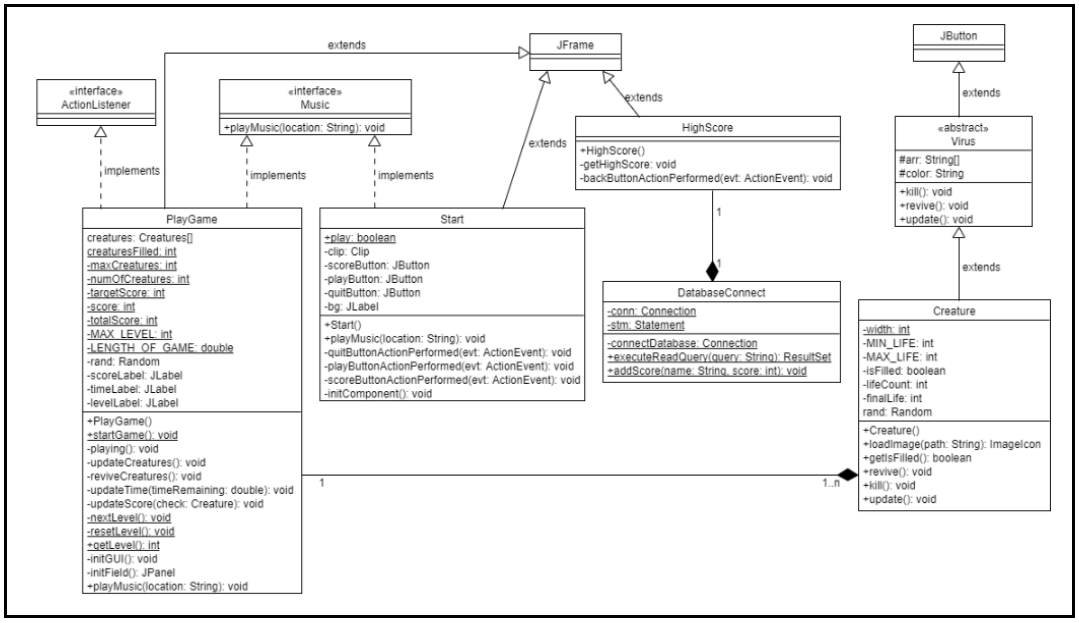
   Kelas ini berfungsi untuk memainkan musik dalam permainan. Adapun *method* yang digunakan dalam kelas ini sebagai berikut:

   a. playMusic(): *Method* ini berfungsi untuk memainkan musik dalam permainan.

8. *Class* ActionListener
9. *Class* JFrame
10. *Class* JButton

BAB III IMPLEMENTASI

3.1 Implementation Class Diagram



Dari diagram kelas pada gambar diatas, dibuat dalam bentuk program Java, dengan jumlah 10 kelas yang memiliki keterhubungan tertentu antar kelasnya.

3.2 Implementasi Konsep Pemrograman Berorientasi Objek

3.2.1 Abstraksi

Abstraksi adalah suatu metode untuk melakukan transformasi dari suatu kasus ke dalam bentuk entitas dan relasinya.

Entitas	Atribut	Operasi
PlayGame	creatures: Creatures[] creaturesFilled: int -maxCreatures: int -numOfCreatures: int -targetScore: int -score: int -totalScore: int -MAX_LEVEL: int -LENGTH_OF_GAME: double -rand: Random -scoreLabel: JLabel -timeLabel: JLabel -levelLabel: Jlabel -timestamp: Timestamp	+PlayGame() +startGame(): void -playing(): void -updateCreatures(): void -reviveCreatures(): void -updateTime(timeRemaining: double): void -updateScore(check: Creature): void -nextLevel(): void -resetLevel(): void +getLevel(): int -initGUI(): void -initField(): JPanel +playMusic(location: String): void +actionPerformed(event: ActionEvent): void
Start	+play: boolean	+Start()

	-clip: Clip -scoreButton: JButton -playButton: JButton -quitButton: JButton -bg: JLabel	+playMusic(location: String): void -quitButtonActionPerformed(evt: ActionEvent): void -playButtonActionPerformed(evt: ActionEvent): void -scoreButtonActionPerformed(evt: ActionEvent): void -initComponent(): void
HighScore		+HighScore() -getHighScore: void -backButtonActionPerformed(evt: ActionEvent): void
DatabaseConnect	-conn: Connection -stm: Statement	-connectDatabase: Connection +executeReadQuery(query: String): ResultSet +addScore(timestamp: Timestamp, name: String, score: int): void
Virus	#arr: String[] #color: String	+kill(): void +revive(): void +update(): void
Creature	-width: int -MIN_LIFE: int -MAX_LIFE: int -isFilled: boolean -lifeCount: int -finalLife: int rand: Random	+Creature() +loadImage(path:String): ImageIcon +getIsFilled(): boolean +revive(): void +kill(): void +update(): void
Music		+playMusic(location: String): void

3.2.2 Enkapsulasi

Enkapsulasi adalah suatu cara untuk menyembunyikan informasi detail dari suatu *class*. Contoh penerapan enkapsulasi pada program "Whack Corona" adalah sebagai berikut:

```
public class PlayGame extends JFrame implements ActionListener,
Music{
    private static int level = 1;

    public static int getLevel(){
            return level;
    }
}
```

Dengan menggunakan *access modifier* berupa "`private`" pada atribut "`level`" akan membuat atribut tersebut tidak dapat diakses dari kelas lain dan hanya bisa diakses dengan menggunakan *method* "`getLevel()`".

3.2.3 Pewarisan (Inheritance)

Pewarisan adalah konsep pemograman di mana sebuah *class* dapat menurunkan properti dan *method* yang dimiliki oleh *class* lainnya. Konsep *inheritance* bisanya digunakan untuk memanfaatkan fitur *code reuse* (penggunaan kembali kode) untuk mengindari duplikasi kode pemograman. Contoh penerapan pewarisan pada program "Whack Corona" adalah sebagai berikut:

```
public class Start extends JFrame{
      public Start() {
           setSize(505, 525);
           setResizable(false);
           setLocationRelativeTo(null);
           setVisible(true);
    }
}
```

Pada kodingan di atas, kelas "`Start`" menjadi kelas turunan dari kelas "`JFrame`" sehingga kelas "`Start`" dapat menggunakan atribut dan *method* yang dimiliki oleh kelas "`JFrame`".

3.2.4 Polymorphism

Polymorphism merupakan kondisi di mana terdapat *method* dengan nama yang sama namun memiliki aksi yang berbeda. Polymorphism dibagi dua, yaitu *overloading* dan *overriding*. *Overloading* adalah *method* dengan nama yang sama, namun memiliki parameter yang berbeda. *Overriding* merupakan method dengan nama yang sama, tipe data yang sama, dan parameter yang sama dengan yang ada pada kelas induk, namun memiliki isi yang berbeda. Contoh penerapan *overriding* pada program "Whack Corona" adalah sebagai berikut:

```
public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}

public class PlayGame extends JFrame implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent event){
        Creature clickedCreature = (Creature) event.getSource();
         Creature clickedCreature = (Creature)event.getSource();
            if(clickedCreature.getIsFilled()){
            clickedCreature.kill();
            updateScore(clickedCreature);
```

```
        }
    }
}
```

Kelas "`PlayGame`" meng-*override* method "`actionPerformed()`" yang dimiliki oleh induknya yaitu "`ActionListener`".

3.2.5 Abstract Class

Abstract class adalah kelas yang mempunyai setidaknya satu *abstract method*. *Abstract method* adalah *method* yang tidak memiliki *body* (hanya deklarasi *method*). Kelas turunan dari suatu *abstract class* harus mengimplementasikan *abstract method* yang yang dimiliki induknya. Contoh penerapan *abstract class* pada program "Whack Corona" adalah sebagai berikut:

```
public abstract class Virus extends JButton {
    protected String[] arr = {"black", "red", "green", "blue"};
    protected String color; // warna virus terpilih

    public abstract void kill();
    public abstract void revive();
    public abstract void update();
}
public class Creature extends Virus {
    public void revive() {
        finalLife = MIN_LIFE + rand.nextInt(MAX_LIFE - MIN_LIFE
+ 1);
        isFilled = true
        color = arr[rand.nextInt(4)];
        if(color.equals("black"))
            this.setIcon(loadImage("Assets/black.png"));
        else if(color.equals("red"))
            this.setIcon(loadImage("Assets/red.png"));
        else if(color.equals("green"))
            this.setIcon(loadImage("Assets/green.png"));
        else
            this.setIcon(loadImage("Assets/blue.png"));
    }

    public void kill() {
      isFilled = false;
      this.setIcon(loadImage("Assets/frame.png"));
      lifeCount = 0;
      PlayGame.creaturesFilled--;
    }

    public void update() {
      if(isFilled){
            lifeCount++;
            if(lifeCount == finalLife)
            this.kill(); // basmi
        }
    }
}
```

Abstract class "`Virus`" memiliki 3 *abstract method*. Sehingga kelas turunannya yaitu kelas "`Creature`" harus mengimplementasikan *method-method* tersebut.

3.2.6 Interface

Interface merupakan wadah yang memiliki atribut yang bersifat *final* dan kumpulan *method* abstrak yang tidak memiliki *body*. Kelas yang mengimplementasikan *interface* tersebut harus mengimplementasikan seluruh *method* abstrak tersebut. Contoh penerapan *interface* pada program "Whack Corona" adalah sebagai berikut:

```
public interface Music {
    public void playMusic(String location);
}
public class Start extends JFrame implements Music {
    public void playMusic(String location){
        try {
            File musicPath = new File(location);
            if(musicPath.exists()) {
                AudioInputStream audioInput =
AudioSystem.getAudioInputStream(musicPath);
                clip = AudioSystem.getClip();
                clip.open(audioInput);
                clip.start();
                clip.loop(Clip.LOOP_CONTINUOUSLY);
            } else {
                System.out.println("Cannot fint the Audio
File");
            }
        } catch (Exception ex){
            ex.printStackTrace();
        }
    }
}
```

Kelas "`Start`" mengimplementasikan method "`playMusic()`" yang didefinisikan pada interface "`Music`".

3.2.7 GUI

Pada program "Whack Corona", GUI dibuat menggunakan *tool pallete* yang ada pada NetBeans IDE. GUI membuat program menjadi lebih atraktif dan interaktif. Kelas yang menerapkan GUI yaitu "`Start`", "`HighScore`", dan "`PlayGame`" yang meng-*extends* "`JFrame`".

3.2.8 Multithreading

Multithreading mengacu kepada dua atau lebih tugas atau *thread* yang berjalan (sedang dieksekusi) di dalam satu program. *Thread* merupakan suatu eksekusi independen di dalam program. Banyak *thread* dapat berjalan secara konkuren di dalam program. *Thread* pada program "Whack Corona" digunakan untuk meng-*update* waktu tersisa pada saat permainan berlangsung.

BAB IV PENUTUP

4.1 Kesimpulan

Terdapat beberapa kesimpulan dari program yang dibuat, yaitu sebagai berikut:

1. Mahasiswa dapat mengimplementasikan konsep Pemrograman Berorientasi Objek pada pembuatan project akhir dari semua modul yang telah dikerjakan saat praktikum.
2. Memberikan hiburan yang lebih menarik serta menjadikan sarana untuk mengusir kepenatan di masa pandemi sekarang.

4.2 Saran

Dalam pembuatan permainan “Whack Corona” ini diharapkan agar tampilan menu dibuat lebih menarik dan *sound effect* diperbaiki kembali agar dapat lebih maksimal.

LAMPIRAN

1. Class Creature

```
package coronawhacks;

import java.awt.Image;
import java.util.Random;
import javax.swing.ImageIcon;

public class Creature extends Virus {
    private int width; // lebar kotak
    private final int MIN_LIFE = 10; // min lama hidup
    private final int MAX_LIFE = 50;  // max lama hidup
    Random rand = new Random(); // objek random
    private boolean isFilled; // kondisi kotak terisi
    private int lifeCount; // lamanya kemunculan kotak
    private int finalLife; //stores how long the creature will be
alive for this life

    public Creature() {
      isFilled = false;
      this.setIcon(loadImage("Assets/frame.png"));
      lifeCount = 0;
    }

    public ImageIcon loadImage(String path){
        // fungsi untuk load gambar
        Image image = new
ImageIcon(this.getClass().getResource(path)).getImage();
        if(PlayGame.getLevel() == 1)
            width = 227;
        else if(PlayGame.getLevel() == 2)
            width = 150;
        else if(PlayGame.getLevel() == 3)
            width = 111;
        else
            width = 88;
        Image scaledImage = image.getScaledInstance(width, 110,
Image.SCALE_SMOOTH);
        return new ImageIcon(scaledImage);
    }

    public boolean getIsFilled() {
      return isFilled; // mengecek apakah terisi
    }

    public void revive() {
        finalLife = MIN_LIFE + rand.nextInt(MAX_LIFE - MIN_LIFE +
1); // lama kemunculan
        isFilled = true; // kotak terisi
        color = arr[rand.nextInt(4)]; // mengacak warna virus
terpilih
        if(color.equals("black"))
            this.setIcon(loadImage("Assets/black.png"));
        else if(color.equals("red"))
            this.setIcon(loadImage("Assets/red.png"));
        else if(color.equals("green"))
            this.setIcon(loadImage("Assets/green.png"));
        else
```

```
            this.setIcon(loadImage("Assets/blue.png"));
    }

    public void kill() {
        // fungsi untuk menghilangkan kotak yang terisi virus
      isFilled = false;
      this.setIcon(loadImage("Assets/frame.png"));
      lifeCount = 0;
      PlayGame.creaturesFilled--;
    }

    public void update() {
        // fungsi untuk update durasi hidup kotak
      if(isFilled){
            lifeCount++;
            // kondisi jika telah mencapai durasi lama kemunculan
            if(lifeCount == finalLife)
            this.kill(); // basmi
        }
    }
}
```

2. Class DatabaseConnect

```
package coronawhacks;

import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;

public class DatabaseConnect {
    private static Connection conn = connectDatabase();
    private static Statement stm;

    private static Connection connectDatabase(){
        try {
            String url ="jdbc:mysql://localhost/whack_corona";
            String user="root";
            String pass="";
            Class.forName("com.mysql.jdbc.Driver");
            conn =DriverManager.getConnection(url,user,pass);
            stm = conn.createStatement();
            System.out.println("Koneksi berhasil.");
        } catch (Exception e) {
            System.err.println("Koneksi Gagal: " +e.getMessage());
        }
        return conn;
    }

    public static ResultSet executeReadQuery(String query) {
      ResultSet rs = null;
      try {
         Statement stt = conn.createStatement();
         rs = stt.executeQuery(query);

      } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return rs;
    }

    public static void addScore(Timestamp ts, String name,int
score){
        try{
            String sql = "insert into player
(started_at,nama,skor) values
("+"'"+ts+"'"+","+"'"+name+"'"+","+"'"+score+"'"+")";
            java.sql.PreparedStatement pstm =
conn.prepareStatement(sql);
            pstm.execute();
        } catch(HeadlessException | SQLException e){
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

3. Class HighScore

```
package coronawhacks;

import java.sql.ResultSet;
import javax.swing.JFrame;
import javax.swing.table.DefaultTableModel;

public class HighScore extends JFrame{
    private javax.swing.JButton backButton;
    private javax.swing.JLabel bg;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tableScore;

    public HighScore() {
        initComponents();
        setSize(505, 525);
        setResizable(false);
        setLocationRelativeTo(null);
        setVisible(true);
        getHighScore();
    }

    private void getHighScore(){
        try{
            ResultSet rs =
DatabaseConnect.executeReadQuery("select * from player order by
skor desc limit 10");
            DefaultTableModel table = (DefaultTableModel)
tableScore.getModel();
            table.setRowCount(0);
            int peringkat[]={1,2,3,4,5,6,7,8,9,10};
            int i=0;
            while(rs.next()){
                table.addRow(new Object[]{ peringkat[i] ,
(rs.getString("nama")), (rs.getInt("skor"))});
                i++;
            }
        } catch (Exception e){
            e.printStackTrace();
```

```
        }
    }

    private void initComponents() {
        backButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        tableScore = new javax.swing.JTable();
        bg = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE
);
        getContentPane().setLayout(null);

        backButton.setIcon(new
javax.swing.ImageIcon("E:\\NGODING\\Java\\pbo\\CoronaWhacks\\src\\
coronawhacks\\Assets\\backButton.png")); // NOI18N
        backButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                backButtonActionPerformed(evt);
            }
        });
        getContentPane().add(backButton);
        backButton.setBounds(30, 420, 160, 49);

        tableScore.setFont(new java.awt.Font("Times New Roman", 0,
14)); // NOI18N
        tableScore.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "NO", "NAME", "SCORE"
            }
        ));
        tableScore.setFocusable(false);
        tableScore.setInheritsPopupMenu(true);
        tableScore.setRowHeight(20);
        jScrollPane1.setViewportView(tableScore);
        if (tableScore.getColumnModel().getColumnCount() > 0) {

tableScore.getColumnModel().getColumn(0).setMaxWidth(30);
        }
        getContentPane().add(jScrollPane1);
        jScrollPane1.setBounds(30, 200, 440, 130);

        bg.setIcon(new
javax.swing.ImageIcon("E:\\NGODING\\Java\\pbo\\CoronaWhacks\\src\\
coronawhacks\\Assets\\bg-hg.png")); // NOI18N
        getContentPane().add(bg);
        bg.setBounds(0, 0, 500, 500);
    }
```

```
    private void
backButtonActionPerformed(java.awt.event.ActionEvent evt) {
        new Start();
        this.dispose();
    }
}
```

4. Interface Music

```
package coronawhacks;

public interface Music {
    public void playMusic(String location);
}
```

5. Class PlayGame

```
package coronawhacks;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.sql.Timestamp;
import java.text.NumberFormat;
import java.util.Date;
import java.util.Random;
import javax.sound.sampled.*;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class PlayGame extends JFrame implements ActionListener,
Music{
    Creature[] creatures; // objek kotak
    static int creaturesFilled; // jumlah kotak yang tidak kosong
dari
    private static int maxCreatures = 4; // jumlah kotak yang
berisi virus
    private static int numOfCreatures = 6; // jumlah kotak awal
    private static int targetScore = 400; // target skor awal
    private static int score; // skor yang sedang dimainkan
    private static int level = 1; // level yang sedang dimainkan
    private static int totalScore = 0; // total skor untuk
disimpan dalam database
    private static final int MAX_LEVEL = 4; // banyak level
    private final double LENGTH_OF_GAME = 15000.0; // durasi game
per level
    private Random rand = new Random(); // membuat objek random
    private JLabel scoreLabel, timeLabel, levelLabel;
    private static Timestamp timestamp;

    public PlayGame(){
        score = 0;
        creaturesFilled = 0;
```

```
        initGUI();
        setVisible(true);
        setResizable(false);
        setLocationRelativeTo(null);
        if(level == 1){
            FloatControl gainControl = (FloatControl)
Start.clip.getControl(FloatControl.Type.MASTER_GAIN);
            gainControl.setValue(-10.0f);
            timestamp = new Timestamp(System.currentTimeMillis());
// ambil waktu mulai main
        }
    }

    public static void startGame(){
        PlayGame this_game = new PlayGame();
        // tampil pesan
        JOptionPane.showMessageDialog(this_game, "Instruksi:
Perhatikan deskripsi poin dari virus.\n"+
                "Tedapat " + MAX_LEVEL + " level. Setiap level
berdurasi 15 detik.\n"+
                "Semoga berhasil.");

        // perulangan sampai game selesai
        while(true){
      // tetap bermain sampai player gagal mencapai target skor
atau level akhir telah diseleaikan
            while(level <= MAX_LEVEL) {
                // pengumuman perihal rincian level
                JOptionPane.showMessageDialog(this_game, "Level "
+ level + "\n"+
                        "Target skor: "+ targetScore + "\n"+
                        "Jumlah kotak: " + numOfCreatures + "\n"+
                        "Tekan OK untuk mulai.");

                this_game.playing(); // mainkan

                // gagal mencapai target skor
                if(score < targetScore) {
                    JOptionPane.showMessageDialog(this_game,
"Skor: " + score +
                            "\nKamu tidak mencapai " + targetScore
+ " poin.  Game Over!");
                    break;
                }

                // lanjut ke level berikutnya
                if(level < MAX_LEVEL)
                    JOptionPane.showMessageDialog(this_game,
"Selamat, kamu berhasil lanjut ke level selanjutnya!");
                nextLevel();

                // mulai level selanjutnya lagi
                if(level <= MAX_LEVEL) {
                    this_game.dispose();
                    totalScore += score;
                    this_game = new PlayGame();
                }
            }

            // jika player telah menyelesaikan level terakhir
```

```
            if(level > MAX_LEVEL)
            JOptionPane.showMessageDialog(this_game, "Selamat,
kamu berhasil menamatkan game ini!");

            // tambah ke database
            totalScore += score;
            String name = "";
            do{
                name = JOptionPane.showInputDialog(this_game,
"Masukkan namamu?");
                if(name == null || name.equals(""))
                    JOptionPane.showMessageDialog(null,
                              "NAMA TIDAK BOLEH KOSONG!!",
                              "PERINGATAN",
                              JOptionPane.WARNING_MESSAGE);
            } while(name == null || name.equals(""));
            DatabaseConnect.addScore(timestamp, name, totalScore);

            // game selesai, dan menanyakan apakah ingin main lagi
            int response =
JOptionPane.showConfirmDialog(this_game, "Total Skor:
"+totalScore+
                "\nTerima kasih telah bermain, "+name+"\nApakah
kamu ingin bermain lagi?",
                "Bermain Kembali?", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);

            // jika memilih tidak maka akan kembali ke menu utama
            if (response == JOptionPane.NO_OPTION || response ==
JOptionPane.CLOSED_OPTION)
                break;

            // jika iya maka akan kembali bermain lagi dari level
1
            resetLevel();
            this_game.dispose();
            this_game = new PlayGame();
      }
      // keluar dari game
      this_game.dispose();
    }

    private void playing() {
      double startTime = new Date().getTime(); // waktu dimulainya
game
      double currentTime; // waktu saat ini yang direnderdari
setiap frame
      double timeRemaining = LENGTH_OF_GAME; // sisa waktu di game

      // bermain selama 15 detik
      while(( LENGTH_OF_GAME - timeRemaining) < LENGTH_OF_GAME){
            reviveCreatures();
            updateCreatures();

            // sleep selama 32 ms untuk membantu update waktu
            try{
            Thread.sleep(32);
            } catch(InterruptedException e){}

            // mengupdate waktu saat ini
```

```
            currentTime = new Date().getTime();

            // menghitung waktu tersisa
            timeRemaining = LENGTH_OF_GAME - (currentTime -
startTime);

            // update waktu
            updateTime(timeRemaining);
        }
    }

    private void updateCreatures(){
        // menambah life_count dari creature, jika sudah mencapai
final_life maka ganti creature tersebut
      for(int x = 0; x < creatures.length; x++)
            creatures[x].update();
    }

    private void reviveCreatures(){
        // mengecek apakah terdapat kotak yang masih kosong dari
kotak yang seharusnya(maxCreatures)
      if(creaturesFilled < maxCreatures){
            // mengacak tempat kemunculan virus
            int randomCreature = rand.nextInt(numOfCreatures);
            // jika tempat teracak kosong, maka tambahkan
            if(!creatures[randomCreature].getIsFilled()) {
            creatures[randomCreature].revive();
            creaturesFilled++;
            }
        }
    }

    private void updateTime(double timeRemaining) {
      // update label waktu
      timeLabel.setText("TIME: " +
NumberFormat.getInstance().format(timeRemaining/1000));
        // jika waktunya mencapai di bawah 5 detik maka warnai
merah
        if(timeRemaining <= 5000.0){
            timeLabel.setForeground(new Color(255, 0, 0));

playMusic("\\build\\classes\\coronawhacks\\Assets\\beep-29.wav");
        }
    }

    private void updateScore(Creature check) {
        // akumulasi skor dengan melihat warna virus
        if(check.color.equals("black"))
            score += 50;
        else if(check.color.equals("red"))
            score -= 5;
        else if(check.color.equals("green"))
            score += 10;
        else
            score -= 30;
        // skor berwarna merah jika kurang dari target dan jika
sudah maka berwarna hijau
        if(score < targetScore)
            scoreLabel.setForeground(new Color(255, 0, 0));
        else
```

```
            scoreLabel.setForeground(new Color(0, 255, 0));
        // update label skor
      scoreLabel.setText("SCORE: " + score);
    }

    private static void nextLevel() {
        level++; // menambah levle
        numOfCreatures += 3; // menambah 3 kotak
        maxCreatures += 1; // menambah batas kotak terisi
        targetScore += 50; // menambah target skor
    }

    private static void resetLevel() {
      // mengulang seperti peraturan awal
      level = 1;
      numOfCreatures = 6;
        targetScore = 400;
        totalScore = 0;
        maxCreatures = 4;
    }

    public static int getLevel(){
        return level;
    }

    private void initGUI(){
        // mengatur JFrame PlayGame
      setSize(505, 525);
      setLayout(null);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setTitle("Whack-a-Mole");

        // menambahkan JPanel
        JPanel contentPane = new JPanel();
        contentPane.setLayout(null);
        setContentPane(contentPane);

        // menambahkan ket skor
        scoreLabel = new JLabel();

scoreLabel.setBorder(javax.swing.BorderFactory.createLineBorder(ne
w java.awt.Color(0, 0, 0)));
        contentPane.add(scoreLabel);
        scoreLabel.setBounds(20, 10, 170, 30);
        scoreLabel.setFont(new Font("Cambria Math", Font.BOLD,
24));

        // menambahkan ket waktu
        timeLabel = new JLabel();

timeLabel.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        contentPane.add(timeLabel);
        timeLabel.setBounds(20, 50, 170, 30);
        timeLabel.setFont(new Font("Cambria Math", Font.BOLD,
24));

        // menambahkan ket level
        levelLabel = new JLabel();
```

```
levelLabel.setBorder(javax.swing.BorderFactory.createLineBorder(ne
w java.awt.Color(0, 0, 0)));
        contentPane.add(levelLabel);
        levelLabel.setBounds(20, 90, 170, 30);
        levelLabel.setFont(new Font("Cambria Math", Font.BOLD,
24));

        // menambahkan deskripsi
        JLabel description = new JLabel();
        description.setIcon(new
javax.swing.ImageIcon(getClass().getResource("Assets/description.p
ng"))); // NOI18N
        contentPane.add(description);
        description.setBounds(200, 10, 280, 110);

        // membuat panel bermain
      JPanel field = initField();
        field.setBounds(15, 140, 460, 340);
      contentPane.add(field);

        // menambahkan background
        JLabel bg = new JLabel();
        bg.setIcon(new
javax.swing.ImageIcon(getClass().getResource("Assets/bg-
game.png")));
        contentPane.add(bg);
        bg.setBounds(0, 0, 500, 500);


field.setCursor(Toolkit.getDefaultToolkit().createCustomCursor(
            new
ImageIcon(this.getClass().getResource("Assets/pointer.png")).getIm
age(),
            new Point(0,0),"custom cursor1"));

        // tampilkan label skor dan level awal
        scoreLabel.setText("SCORE: " + score);
        levelLabel.setText("LEVEL: " + level);
    }

    private JPanel initField() {
      // membuat panel untuk ruang bermain
      JPanel field = new JPanel();
      field.setLayout(new GridLayout(3, 3, 5, 5));
        field.setOpaque(false); // supaya transparan terhadap
background
      // inisialisasi kotak berisi virus
      creatures = new Creature[numOfCreatures];
      for(int x = 0; x < creatures.length; x++) {
            creatures[x] = new Creature();
            creatures[x].addActionListener(this);
            field.add(creatures[x]);
        }
        return field;
    }

    public void playMusic(String location) {
      try {
            File musicPath = new
File(System.getProperty("user.dir")+location);
```

```
            if(musicPath.exists()) {
                AudioInputStream audioInput =
AudioSystem.getAudioInputStream(musicPath);
                Clip clip = AudioSystem.getClip();
                clip.open(audioInput);
                clip.start();
            } else {
                System.out.println("Cannot fint the Audio File");
            }
        } catch (Exception ex){
            ex.printStackTrace();
        }
    }

    public void actionPerformed(ActionEvent event){
        // jika pemain klik creature yang terisi maka poin
didapatkan
      Creature clickedCreature = (Creature) event.getSource();
      if(clickedCreature.getIsFilled()){

playMusic("\\build\\classes\\coronawhacks\\Assets\\spray.wav");
            clickedCreature.kill();
            updateScore(clickedCreature);
        }
    }
}
```

6. Class Start

```
package coronawhacks;

import java.io.File;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.FloatControl;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Start extends JFrame implements Music {
    private JButton ScoreButton = new JButton();
    private JLabel bg = new JLabel();
    private JButton playButton = new JButton();
    private JButton quitButton = new JButton();
    public static boolean play;
    public static Clip clip;

    public Start() {
        initComponents();
        setSize(505, 525);
        setResizable(false);
        setLocationRelativeTo(null);
        setVisible(true);
        play = false;
    }

    public void playMusic(String location){
        try {
            File musicPath = new
```

```
File(System.getProperty("user.dir")+location);
            if(musicPath.exists()) {
                AudioInputStream audioInput =
AudioSystem.getAudioInputStream(musicPath);
                clip = AudioSystem.getClip();
                clip.open(audioInput);
                clip.start();
                clip.loop(Clip.LOOP_CONTINUOUSLY);
            } else {
                System.out.println("Cannot fint the Audio File");
            }
        } catch (Exception ex){
            ex.printStackTrace();
        }
    }

    private void
quitButtonActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void
playButtonActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
        play = true;
    }

    private void
ScoreButtonActionPerformed(java.awt.event.ActionEvent evt) {
        new HighScore();
        this.dispose();
    }

    private void initComponents() {

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE
);
        setMinimumSize(null);
        getContentPane().setLayout(null);

        quitButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("Assets/quitButton.pn
g"))); // NOI18N
        quitButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        quitButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                quitButtonActionPerformed(evt);
            }
        });
        getContentPane().add(quitButton);
        quitButton.setBounds(170, 360, 190, 70);

        playButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("Assets/playButton.pn
g"))); // NOI18N
        playButton.setBorderPainted(false);
        playButton.setCursor(new
```

```
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        playButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                playButtonActionPerformed(evt);
            }
        });
        getContentPane().add(playButton);
        playButton.setBounds(170, 160, 190, 70);

        ScoreButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("Assets/scoreButton.p
ng"))); // NOI18N
        ScoreButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        ScoreButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                ScoreButtonActionPerformed(evt);
            }
        });
        getContentPane().add(ScoreButton);
        ScoreButton.setBounds(170, 260, 190, 70);

        bg.setIcon(new
javax.swing.ImageIcon(getClass().getResource("Assets/bg-
menu.png"))); // NOI18N
        getContentPane().add(bg);
        bg.setBounds(0, 0, 500, 500);
    }

    public static void main(String[] args){
        new
Start().playMusic("\\build\\classes\\coronawhacks\\Assets\\Carniva
l-Games.wav");
        while (play != true) {
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {}
        }
        PlayGame.startGame();
    }
}
```

7. Abstract Class Virus

```
package coronawhacks;

import javax.swing.JButton;

public abstract class Virus extends JButton {
    protected String[] arr = {"black", "red", "green", "blue"}; //
kumpulan virus
    protected String color; // warna virus terpilih
    public abstract void kill();
    public abstract void revive();
    public abstract void update();
}
```

DAFTAR PUSTAKA

[1] A. S. dan R. Rahmad, “No Title,” *mongabay.co.id*, 2020. https://www.mongabay.co.id/2020/01/28/virus-corona-mewabah-di-wuhan-menyebar-cepat-ke-penjuru-dunia/ (accessed Dec. 13, 2020).

[2] F. Firdaus, “No Title,” *tirto.id*, 2020. https://tirto.id/update-corona-indonesia-dunia-10-mei-2020-data-kasus-terbaru-fpp6 (accessed Dec. 13, 2020).