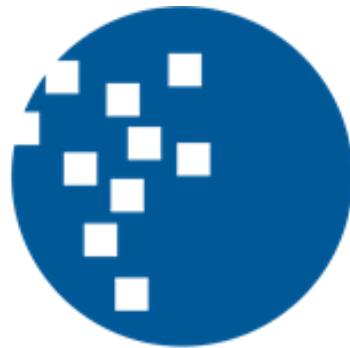


**OBJECT ORIENTED PROGRAMMING
KEBUN BINATANG**



**JONATHAN CAHYO WISANG P - 00000100309
TRISTAN GAUTAMA - 00000099564
FERNANDO - 00000099540
RAMA HARRISON - 00000100226**

**FAKULTAS TEKNIK DAN INFORMATIKA
PROGRAM STUDI SISTEM INFORMASI
UNIVERSITAS MULTIMEDIA NUSANTARA**

2024

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
LATAR BELAKANG.....	3
BAB II.....	5
LANDASAN.....	5
2.1 Tujuan.....	5
2.2 Struktur Kode.....	5
2.3 Pengimplementasian Modul OOP.....	6
BAB III.....	7
HASIL.....	7
BAB IV.....	26
PETUNJUK PENGGUNAAN.....	26
BAB V.....	37
KESIMPULAN.....	37
DAFTAR PUSTAKA.....	41

BAB 1

LATAR BELAKANG

[1] Seiring dengan berkembangnya teknologi dan kompleksitas aplikasi, paradigma pemrograman telah mengalami evolusi dari prosedural menjadi berorientasi objek (Object Oriented Programming, OOP). OOP memungkinkan pengembang untuk menciptakan sistem yang lebih modular, terstruktur, dan mudah dipelihara. Konsep-konsep utama dalam OOP seperti *encapsulation, inheritance, interface, abstraction, polymorphism, polimorphism, arraylist, single linked list, double linked list, circular linked list, double circular linked list, stack, dan queue.*

[2] Kebun binatang memainkan peran penting dalam konservasi, pendidikan, penelitian, dan rekreasi. Sebagai tempat yang menampung berbagai spesies hewan, kebun binatang harus memastikan bahwa setiap hewan mendapatkan perawatan yang tepat sesuai dengan kebutuhannya. Selain itu, kebun binatang juga harus menyediakan habitat yang aman dan nyaman, serta mengelola interaksi dengan pengunjung secara efisien. Kompleksitas tugas ini menuntut adanya sistem informasi yang mampu mengelola berbagai aspek operasional secara efektif. Banyak kebun binatang yang masih mengandalkan sistem manual untuk mencatat informasi tentang hewan dan operasional sehari-hari. Sistem manual ini rentan terhadap kesalahan manusia, memerlukan waktu dan sumber daya yang besar, serta kurang efisien dalam pengelolaan data. Dengan adanya sistem informasi terkomputerisasi yang berbasis OOP, kebun binatang dapat mengelola data dengan lebih efisien, mengurangi risiko kesalahan, dan mempermudah akses serta pemrosesan informasi. Sistem ini dapat disesuaikan dengan kebutuhan spesifik setiap kebun binatang, sehingga mendukung operasional yang lebih optimal.

Proyek akhir ini bertujuan untuk mengembangkan sebuah program berbasis OOP yang dapat digunakan untuk mengelola kebun binatang. Program ini akan mencakup fitur-fitur untuk mengatur data hewan, kandang, jadwal pemberian makan, dan berbagai aspek lain dari manajemen kebun binatang. Pendekatan OOP akan digunakan dalam desain dan implementasi program ini, dengan memanfaatkan prinsip-prinsip seperti *encapsulation, inheritance, interface, abstraction, polymorphism, polimorphism, arraylist, single linked list, double linked list, circular linked list, double circular linked list, stack, dan queue.* Dengan

pendekatan ini, program akan dirancang agar modular, dapat digunakan kembali, dan mudah dipelihara.

Penggunaan OOP dalam pengembangan program kebun binatang tidak hanya memiliki relevansi akademis, tetapi juga aplikasi praktis yang nyata. Program ini dapat diadaptasi dan digunakan oleh kebun binatang untuk meningkatkan efisiensi operasional mereka. Dengan adanya program ini, diharapkan kebun binatang dapat mengelola data hewan dan operasional sehari-hari dengan lebih baik, yang pada akhirnya akan meningkatkan kualitas perawatan hewan dan pengalaman pengunjung. Dampak positif ini mencakup peningkatan efisiensi pengelolaan, pengurangan kesalahan, dan pemanfaatan sumber daya yang lebih baik. Dengan latar belakang ini, proyek akhir pengembangan program kebun binatang berbasis OOP ini diharapkan dapat memberikan solusi yang efektif dan aplikatif dalam pengelolaan kebun binatang, serta menunjukkan penerapan yang nyata dari konsep-konsep OOP dalam dunia nyata.

BAB II

LANDASAN

2.1 Tujuan

Kebun Binatang ini adalah proyek Object Oriented Programming (OOP) yang mengimplementasikan konsep Object Oriented Programming (OOP) dengan menggunakan berbagai modul, seperti encapsulation, inheritance, interface, abstraction, polymorphism, ArrayList, Single Linked List, Double Linked List, Circular Linked List, Double Circular Linked List, Stack, dan Queue. Dengan Tujuan untuk mengembangkan Program yang dapat memasukan daftar hewan, deskripsi hewan, melihat kegiatan hewan, membuat list pengunjung dan lainnya. Program ini juga dilengkapi dengan fitur undo dan antrian pengunjung.

2.2 Manfaat

Kebun Binatang OOP, dengan implementasi konsep OOP yang menyeluruh, menawarkan berbagai manfaat bagi penggunanya, baik dari sisi pengembangan program maupun dari sisi pengalaman pengguna. Bagi Pengembang Program, modul-modul kecil yang terstruktur dapat memudahkan pengembangan, pengujian, dan pemeliharaan program. Bagi pengguna, Kita dapat menggunakan fitur fitur yang menarik dan informatif, seperti daftar hewan, deskripsi hewan, kegiatan hewan, list pengunjung, fitur undo, dan antrian pengunjung.

2.3 Struktur Kode

- Kelas Main: Class utama yang mengatur jalannya program.
- Kelas Hewan: Class untuk mengatur hewan, termasuk nama, deskripsi, dan kegiatan.
- Kelas Karnivora: Class untuk mengatur hewan karnivora.
- Kelas Herbivora: Class untuk mengatur hewan herbivora.
- Kelas Pengunjung: Class untuk mengatur pengunjung, termasuk nama dan informasi lainnya.
- Kelas UndoAction: Class untuk mengatur tindakan yang dapat di-undo.
- Kelas VisitorQueue: Class untuk mengatur antrian pengunjung.

- Kelas ZooInterface: Class untuk mengatur interface yang harus diimplementasikan oleh class ZooKeeper.
- Kelas ZooKeeper: Class untuk mengatur kegiatan di kebun binatang, termasuk pengelolaan hewan, pengelolaan pengunjung, dan fitur undo.
- Kelas CircularList: Class untuk mengatur linked list yang membentuk lingkaran.
- Kelas DoubleCircularLinkedList: Class untuk mengatur linked list yang memiliki dua pointer dan membentuk lingkaran.

2.4 Pengimplementasian Modul OOP

- Encapsulation: Menggunakan getter dan setter untuk mengatur akses data, seperti nama di class Pengunjung dan Hewan.
- Inheritance: Menurunkan class Hewan ke Karnivora dan dari class Karnivora ke Herbivora menggunakan kata kunci "extends" dan anotasi "@Override".
- Interface: Menggunakan interface untuk menentukan metode yang harus diimplementasikan oleh class ZooKeeper.
- Abstraction: Menggunakan class abstrak untuk mengatur perilaku objek tanpa memperhatikan bagaimana objek tersebut dilakukan.
- Polymorphism: Menggunakan metode yang sama untuk dipanggil pada objek yang berbeda, menghasilkan perilaku yang berbeda tergantung pada jenis objek yang memanggil metode tersebut.
- ArrayList: Menggunakan array untuk menyimpan objek Hewan.
- Single Linked List: Menggunakan linked list untuk mengantri pengunjung.
- Double Linked List: Menggunakan linked list yang memiliki dua pointer, satu untuk node berikutnya dan satu untuk node sebelumnya.
- Circular Linked List: Menggunakan linked list yang membentuk lingkaran.
- Double Circular Linked List: Menggunakan kombinasi dari double linked list dan circular linked list.
- Stack: Menggunakan stack untuk undo actions.
- Queue: Menggunakan metode enqueue dan dequeue untuk mengantri dan mengeluarkan pengunjung dari antrian.

BAB III

HASIL

1. Kelas Main

```
● ● ●

1 package KebunBinatang;
2
3 import java.util.InputMismatchException;
4 import java.util.Scanner;
5
6 public class main {
7     static Hewan[] hewanArray = new Hewan[10];
8     static int hewanCount = 0;
9     static int currentIndex = 0;
10    static UndoAction[] undoStack = new UndoAction[10];
11    static int top = -1;
12
13    public static void main(String[] args) {
14        Scanner scanner = new Scanner(System.in);
15        ZooKeeper zooKeeper = new ZooKeeper();
16        VisitorQueue visitorQueue = new VisitorQueue();
17
18        while (true) {
19            System.out.println("\nMenu:");
20            System.out.println("1. Tambah Hewan");
21            System.out.println("2. Tampilkan Hewan");
22            System.out.println("3. Swipe Hewan");
23            System.out.println("4. Swap Hewan");
24            System.out.println("5. Hapus Hewan");
25            System.out.println("6. Batal Tambah Hewan");
26            System.out.println("7. Tambah Pengunjung");
27            System.out.println("8. Tampilkan Pengunjung");
28            System.out.println("9. Pengunjung Masuk");
29            System.out.println("10. Keluar");
30            System.out.print("Pilih menu: ");
```

```
1 try {
2         int pilihan = scanner.nextInt();
3         scanner.nextLine();
4
5         switch (pilihan) {
6             case 1:
7                 zooKeeper.tambahHewan(scanner);
8                 break;
9             case 2:
10                zooKeeper.tampilkanHewan();
11                break;
12            case 3:
13                zooKeeper.swipeHewan(scanner);
14                break;
15            case 4:
16                zooKeeper.swapHewan(scanner);
17                break;
18            case 5:
19                zooKeeper.deleteHewan(scanner);
20                break;
21            case 6:
22                zooKeeper.undo();
23                break;
24            case 7:
25                visitorQueue.tambahPengunjung(scanner);
26                break;
27            case 8:
28                visitorQueue.tampilkanPengunjung();
29                break;
30            case 9:
31                visitorQueue.visitZoo();
32                break;
33            case 10:
34                System.out.println("Terima kasih sudah berkunjung ke kebun binatang.");
35                scanner.close();
36                return;
37            default:
38                System.out.println("Pilihan tidak valid.");
39        }
40    } catch (InputMismatchException e) {
41        System.out.println("Input tidak valid. Harap masukkan angka.");
42        scanner.nextLine();
43    }
44}
45}
46
47 static boolean isEmpty() {
48     return (top == -1);
49 }
50 }
```

2. Kelas Hewan

```
● ● ●  
1 package KebunBinatang;  
2  
3 abstract class Hewan {  
4     protected String namahewan;  
5     protected String makanan;  
6     protected String habitat;  
7     protected String deskripsi;  
8     protected String suara;  
9     protected String kegiatan;  
10  
11     public Hewan(String namahewan, String makanan, String habitat, String deskripsi, String kegiatan, String suara) {  
12         this.namahewan = namahewan;  
13         this.makanan = makanan;  
14         this.habitat = habitat;  
15         this.deskripsi = deskripsi;  
16         this.kegiatan = kegiatan;  
17         this.suara = suara;  
18     }  
19  
20     public String getNamahewan() {  
21         return namahewan;  
22     }  
23  
24     public void setNamahewan(String namahewan) {  
25         this.namahewan = namahewan;  
26     }  
27  
28     public String getMakanan() {  
29         return makanan;  
30     }  
31  
32     public void setMakanan(String makanan) {  
33         this.makanan = makanan;  
34     }  
35  
36     public String getHabitat() {  
37         return habitat;  
38     }  
39  
40     public void setHabitat(String habitat) {  
41         this.habitat = habitat;  
42     }  
43  
44     public void setDeskripsi(String deskripsi) {  
45         this.deskripsi = deskripsi;  
46     }  
47  
48     public void setSuara(String suara) {  
49         this.suara = suara;  
50     }  
51  
52     public void setKegiatan(String kegiatan) {  
53         this.kegiatan = kegiatan;  
54     }  
55  
56     public abstract void info();  
57  
58     public abstract void deskripsi();  
59  
60     public abstract void suara();  
61  
62     public abstract void kegiatan();  
63 }  
64 }
```

3. Kelas Karnivora

```
● ● ●  
1 package KebunBinatang;  
2  
3 class Karnivora extends Hewan {  
4     public Karnivora(String namahewan, String makanan, String habitat, String deskripsi, String suara,  
5                         String kegiatan) {  
6         super(namahewan, makanan, habitat, deskripsi, suara, kegiatan);  
7     }  
8  
9     @Override  
10    public void info() {  
11        System.out.println("Nama Hewan: " + namahewan);  
12        System.out.println("Jenis Makanan: " + makanan);  
13        System.out.println("Habitat: " + habitat);  
14    }  
15  
16    public void deskripsi() {  
17        System.out.println("Deskripsi dari " + namahewan + ": " + deskripsi);  
18    }  
19  
20    public void suara() {  
21        System.out.println("Suara dari " + namahewan + ": " + suara);  
22    }  
23  
24    public void kegiatan() {  
25        System.out.println("Kegiatan dari " + namahewan + ": " + kegiatan);  
26    }  
27 }
```

4. Kelas Herbivora

```
● ● ●  
1 package KebunBinatang;  
2  
3 class Herbivora extends Karnivora {  
4     public Herbivora(String namahewan, String makanan, String habitat, String deskripsi, String suara,  
5                         String kegiatan) {  
6         super(namahewan, makanan, habitat, deskripsi, suara, kegiatan);  
7     }  
8 }
```

5. Kelas Pengunjung



```
1 package KebunBinatang;
2
3 class Pengunjung {
4     private String nama;
5     private int umur;
6     private String asal;
7     private String jenisKelamin;
8     private String hewanFavorit;
9
10    public Pengunjung(String nama, int umur, String asal, String jenisKelamin, String hewanFavorit) {
11        this.nama = nama;
12        this.umur = umur;
13        this.asal = asal;
14        this.jenisKelamin = jenisKelamin;
15        this.hewanFavorit = hewanFavorit;
16    }
17
18    public String getNama() {
19        return nama;
20    }
21
22    public void setNama(String nama) {
23        this.nama = nama;
24    }
25
26    public int getUmur() {
27        return umur;
28    }
29
30    public void setUmur(int umur) {
31        this.umur = umur;
32    }
33
34    public String getAsal() {
35        return asal;
36    }
37
38    public void setAsal(String asal) {
39        this.asal = asal;
40    }
41
42    public String getJenisKelamin() {
43        return jenisKelamin;
44    }
45
46    public void setJenisKelamin(String jenisKelamin) {
47        this.jenisKelamin = jenisKelamin;
48    }
49
50    public String getHewanFavorit() {
51        return hewanFavorit;
52    }
53
54    public void setHewanFavorit(String hewanFavorit) {
55        this.hewanFavorit = hewanFavorit;
56    }
57
58    public void info() {
59        System.out.println();
60        System.out.println("Pengunjung ini bernama " + nama);
61        System.out.println(nama + " berasal dari " + asal + " dan berusia " + umur + " tahun");
62    }
63
64    public void HewanFavorit() {
65        System.out.println("hewan favorit " + nama + " adalah " + hewanFavorit);
66    }
67 }
```

6. Kelas UndoAction



```
1 package KebunBinatang;
2
3 class UndoAction {
4     enum Type {
5         ADD, SWIPE, SWAP
6     }
7
8     private Type type;
9     private int index;
10    private int index1;
11    private int index2;
12
13    UndoAction(Type type, int index) {
14        this.type = type;
15        this.index = index;
16    }
17
18    UndoAction(Type type, int index1, int index2) {
19        this.type = type;
20        this.index1 = index1;
21        this.index2 = index2;
22    }
23
24    public Type getType() {
25        return type;
26    }
27
28    public int getIndex() {
29        return index;
30    }
31
32    public int getIndex1() {
33        return index1;
34    }
35
36    public int getIndex2() {
37        return index2;
38    }
39}
40
```

7. Kelas VisitorQueue



```
1 package KebunBinatang;
2
3 import java.util.Scanner;
4
5 class VisitorQueue {
6     private Node head;
7     private Node tail;
8
9     public VisitorQueue() {
10         head = null;
11         tail = null;
12     }
13
14     public void tambahPengunjung(Scanner scanner) {
15         System.out.print("Nama pengunjung: ");
16         String nama = scanner.nextLine();
17         System.out.print("Umur pengunjung: ");
18         int umur = scanner.nextInt();
19         scanner.nextLine();
20         System.out.print("Asal pengunjung: ");
21         String asal = scanner.nextLine();
22         System.out.print("Jenis kelamin pengunjung: ");
23         String jenisKelamin = scanner.nextLine();
24         System.out.print("Hewan favorit pengunjung: ");
25         String hewanFavorit = scanner.nextLine();
26
27         Pengunjung pengunjung = new Pengunjung(nama, umur, asal, jenisKelamin, hewanFavorit);
28
29         if (isInQueue(pengunjung.getName())) {
30             System.out.println("Pengunjung " + pengunjung.getName() + " sudah berada di antrian.");
31             return;
32         }
33
34         enqueue(pengunjung);
35         System.out.println("Pengunjung berhasil ditambahkan ke antrian.");
36
37         if (!isEmpty()) {
38             System.out
39                 .println("\nPengunjung saat ini sedang mengunjungi Kebun Binatang: " + head.data.getName());
40             System.out.println("Pengunjung " + pengunjung.getName() + " sedang menunggu dalam antrian.");
41         }
42     }
43 }
```



```
1  public void tampilanPengunjung() {
2      if (isEmpty()) {
3          System.out.println("Tidak ada pengunjung dalam antrian.");
4          return;
5      }
6
7      System.out.println("\nAntrian Pengunjung:");
8      Node current = head;
9      int i = 1;
10     while (current != null) {
11         System.out.println(i + ". " + current.data.getNama());
12         current = current.next;
13         i++;
14     }
15 }
16
17 public void enqueue(Pengunjung pengunjung) {
18     Node newNode = new Node(pengunjung);
19     if (isEmpty()) {
20         head = newNode;
21         tail = newNode;
22     } else {
23         tail.next = newNode;
24         tail = newNode;
25     }
26 }
27
28 public Pengunjung dequeue() {
29     if (isEmpty()) {
30         return null;
31     }
32     Pengunjung temp = head.data;
33     head = head.next;
34     if (head == null) {
35         tail = null;
36     }
37     return temp;
38 }
```



```
1  public boolean isEmpty() {
2      return (head == null);
3  }
4
5  public void visitZoo() {
6      if (!isEmpty()) {
7          Pengunjung currentVisitor = dequeue();
8          System.out.println(
9              "\nPengunjung " + currentVisitor.getNama() + " sekarang sedang mengunjungi Kebun Binatang.");
10     } else {
11         System.out.println("\nTidak ada orang dalam antrian.");
12     }
13 }
14
15 private class Node {
16     private Pengunjung data;
17     private Node next;
18
19     public Node(Pengunjung data) {
20         this.data = data;
21         this.next = null;
22     }
23 }
24
25 private boolean isInQueue(String visitorName) {
26     Node current = head;
27     while (current != null) {
28         if (current.data.getNama().equals(visitorName)) {
29             return true;
30         }
31         current = current.next;
32     }
33     return false;
34 }
35 }
```

8. Kelas ZooInterface

```
1 package KebunBinatang;  
2  
3 import java.util.Scanner;  
4  
5 interface ZooInterface {  
6     void tambahHewan(Scanner scanner);  
7  
8     void tampilkanHewan();  
9  
10    void swipeHewan(Scanner scanner);  
11  
12    void swapHewan(Scanner scanner);  
13  
14    void undo();  
15  
16    void deleteHewan(Scanner scanner);  
17 }  
18
```

9. Kelas ZooKeeper



```
1 package KebunBinatang;
2
3 import java.util.Scanner;
4
5 class ZooKeeper implements ZooInterface {
6     Hewan[] hewanArray = new Hewan[10];
7     int hewanCount = 0;
8     int currentIndex = 0;
9     UndoAction[] undoStack = new UndoAction[10];
10    int top = -1;
11
12    @Override
13    public void tampilkanHewan() {
14        if (hewanCount == 0) {
15            System.out.println("Tidak ada hewan di dalam daftar.");
16            return;
17        }
18
19        System.out.println("\nDaftar Hewan:");
20        for (int i = 0; i < hewanCount; i++) {
21            System.out.println((i + 1) + ". " + hewanArray[i].getNamahewan());
22        }
23    }
24}
```



```
1  @Override
2      public void swipeHewan(Scanner scanner) {
3          if (hewanCount == 0) {
4              System.out.println("Tidak ada hewan di dalam daftar.");
5              return;
6          }
7
8          System.out.println("\nInformasi Hewan Saat Ini:");
9          hewanArray[currentIndex].info();
10         hewanArray[currentIndex].deskripsi();
11         hewanArray[currentIndex].suara();
12         hewanArray[currentIndex].kegiatan();
13
14         while (true) {
15             System.out.println("\nSwipe Hewan:");
16             System.out.println("1. Swipe ke kanan (Next)");
17             System.out.println("2. Swipe ke kiri (Previous)");
18             System.out.println("3. Kembali ke Menu Utama");
19             System.out.print("Pilih aksi: ");
20
21             int aksi = scanner.nextInt();
22             scanner.nextLine();
23
24             if (aksi == 1) {
25                 currentIndex = (currentIndex + 1) % hewanCount;
26                 System.out.println("\nInformasi Hewan:");
27                 hewanArray[currentIndex].info();
28                 hewanArray[currentIndex].deskripsi();
29                 hewanArray[currentIndex].suara();
30                 hewanArray[currentIndex].kegiatan();
31             } else if (aksi == 2) {
32                 currentIndex = (currentIndex - 1 + hewanCount) % hewanCount;
33                 System.out.println("\nInformasi Hewan:");
34                 hewanArray[currentIndex].info();
35                 hewanArray[currentIndex].deskripsi();
36                 hewanArray[currentIndex].suara();
37                 hewanArray[currentIndex].kegiatan();
38             } else if (aksi == 3) {
39                 break;
40             } else {
41                 System.out.println("Aksi tidak valid.");
42             }
43
44             push(new UndoAction(UndoAction.Type.SWIPE, currentIndex));
45         }
46     }
47 }
```

```

● ● ●

1  @Override
2      public void undo() {
3          if (isEmpty()) {
4              System.out.println("Tidak ada aksi yang dapat di-undo.");
5              return;
6          }
7
8          UndoAction action = undoStack[top];
9          top--;
10
11         if (action.getType() == UndoAction.Type.ADD) {
12             hewanCount--;
13             hewanArray[hewanCount] = null;
14         } else if (action.getType() == UndoAction.Type.SWIPE) {
15             currentIndex = action.getIndex();
16         } else if (action.getType() == UndoAction.Type.SWAP) {
17             Hewan temp = hewanArray[action.getIndex1()];
18             hewanArray[action.getIndex1()] = hewanArray[action.getIndex2()];
19             hewanArray[action.getIndex2()] = temp;
20         }
21
22         System.out.println("Aksi berhasil di-undo.");
23     }
24
25     @Override
26     public void swapHewan(Scanner scanner) {
27         if (hewanCount < 2) {
28             System.out.println("Tidak cukup hewan untuk di-swap.");
29             return;
30         }
31
32         tampilkanHewan();
33
34         System.out.println("\nSwap Hewan:");
35         System.out.print("Masukkan urutan hewan pertama: ");
36         int index1 = scanner.nextInt() - 1;
37         scanner.nextLine();
38
39         System.out.print("Masukkan urutan hewan kedua: ");
40         int index2 = scanner.nextInt() - 1;
41         scanner.nextLine();
42
43         if (index1 < 0 || index1 >= hewanCount || index2 < 0 || index2 >= hewanCount) {
44             System.out.println("Indeks hewan tidak valid.");
45             return;
46         }
47
48         Hewan temp = hewanArray[index1];
49         hewanArray[index1] = hewanArray[index2];
50         hewanArray[index2] = temp;
51
52         push(new UndoAction(UndoAction.Type.SWAP, index1, index2));
53
54         System.out.println("Hewan berhasil di-swap!");
55     }

```

```
● ● ●  
1  Override  
2      public void deleteHewan(Scanner scanner) {  
3          if (hewanCount == 0) {  
4              System.out.println("Tidak ada hewan di dalam daftar.");  
5              return;  
6          }  
7  
8          tampilanHewan();  
9  
10         System.out.print("\nMasukkan urutan hewan yang ingin dihapus: ");  
11         int index = scanner.nextInt() - 1;  
12         scanner.nextLine();  
13  
14         if (index < 0 || index >= hewanCount) {  
15             System.out.println("Indeks hewan tidak valid.");  
16             return;  
17         }  
18  
19         for (int i = index; i < hewanCount - 1; i++) {  
20             hewanArray[i] = hewanArray[i + 1];  
21         }  
22  
23         hewanCount--;  
24         hewanArray[hewanCount] = null;  
25  
26         if (currentIndex >= index) {  
27             currentIndex = (currentIndex - 1 + hewanCount) % hewanCount;  
28         }  
29  
30         push(new UndoAction(UndoAction.Type.ADD, index));  
31  
32         System.out.println("Hewan berhasil dihapus!");  
33     }
```

```

1  @Override
2  public void tambahHewan(Scanner scanner) {
3      System.out.println("\nJenis Hewan:");
4      System.out.println("- Herbivora");
5      System.out.println("- Karnivora");
6      System.out.print("Pilih jenis hewan: ");
7
8      String jenisHewanInput = scanner.nextLine().trim().toLowerCase();
9
10     int jenisHewan = 0;
11     if (jenisHewanInput.equals("herbivora")) {
12         jenisHewan = 1;
13     } else if (jenisHewanInput.equals("karnivora")) {
14         jenisHewan = 2;
15     } else {
16         System.out.println("Jenis hewan tidak valid. Masukkan 'Herbivora' atau 'Karnivora'.");
17         return;
18     }
19
20     System.out.print("Nama hewan: ");
21     String namaHewan = scanner.nextLine();
22     System.out.print("Makanan: ");
23     String makanan = scanner.nextLine();
24     System.out.print("Habitat: ");
25     String habitat = scanner.nextLine();
26     System.out.print("Deskripsi: ");
27     String deskripsi = scanner.nextLine();
28     System.out.print("Suara: ");
29     String suara = scanner.nextLine();
30     System.out.print("Kegiatan: ");
31     String kegiatan = scanner.nextLine();
32
33     Hewan hewan;
34
35     if (jenisHewan == 1) {
36         hewan = new Herbivora(namaHewan, makanan, habitat, deskripsi, suara, kegiatan);
37     } else if (jenisHewan == 2) {
38         hewan = new Karnivora(namaHewan, makanan, habitat, deskripsi, suara, kegiatan);
39     } else {
40         System.out.println("Jenis hewan tidak valid.");
41         return;
42     }
43
44     hewanArray[hewanCount] = hewan;
45     hewanCount++;
46
47     push(new UndoAction(UndoAction.Type.ADD, hewanCount - 1));
48
49     System.out.println("Hewan berhasil ditambahkan!");
50 }
51
52 boolean isEmpty() {
53     return (top == -1);
54 }
55
56 void push(UndoAction action) {
57     if (isFull()) {
58
59         for (int i = 0; i < top; i++) {
60             undoStack[i] = undoStack[i + 1];
61         }
62         top--;
63     }
64
65     top++;
66     undoStack[top] = action;
67 }
68
69 boolean isFull() {
70     return (top == undoStack.length - 1);
71 }
72 }

```

10.Kelas CircularList



```
1 package KebunBinatang;
2
3 class CircularAnimalList {
4     private Node head;
5
6     private class Node {
7         private Hewan data;
8         private Node next;
9
10    public Node(Hewan data) {
11        this.data = data;
12        this.next = null;
13    }
14 }
15
16 public CircularAnimalList() {
17     head = null;
18 }
19
20 public void addAnimal(Hewan animal) {
21     Node newNode = new Node(animal);
22     if (head == null) {
23         head = newNode;
24         newNode.next = head;
25     } else {
26         Node current = head;
27         while (current.next != head) {
28             current = current.next;
29         }
30         current.next = newNode;
31         newNode.next = head;
32     }
33 }
34
35 public void displayAnimals() {
36     if (head == null) {
37         System.out.println("No animals in the list.");
38         return;
39     }
40
41     Node current = head;
42     do {
43         System.out.println(current.data.getNamahewan());
44         current = current.next;
45     } while (current != head);
46 }
```



```
1     public Hewan removeAnimal(String name) {
2         if (head == null) {
3             return null;
4         }
5
6         if (head.data.getNamahewan().equals(name)) {
7             if (head.next == head) { // Only one node
8                 head = null;
9             } else {
10                 Node current = head;
11                 while (current.next != head) {
12                     current = current.next;
13                 }
14                 head = head.next;
15                 current.next = head;
16             }
17             return head.data;
18         }
19
20         Node current = head;
21         Node previous = null;
22         do {
23             if (current.data.getNamahewan().equals(name)) {
24                 if (previous != null) {
25                     previous.next = current.next;
26                 } else {
27                     head = current.next;
28                 }
29                 return current.data;
30             }
31             previous = current;
32             current = current.next;
33         } while (current != head);
34
35         return null;
36     }
37 }
```

11. Kelas DoubleCircularLinkedList

```
● ● ●  
1 package KebunBinatang;  
2  
3 public class DoubleCircularLinkedList {  
4     private Node head;  
5     private Node tail;  
6  
7     private class Node {  
8         Pengunjung data;  
9         Node next;  
10        Node prev;  
11    }  
12    public Node(Pengunjung data) {  
13        this.data = data;  
14        this.next = null;  
15        this.prev = null;  
16    }  
17}  
18  
19    public DoubleCircularLinkedList() {  
20        head = null;  
21        tail = null;  
22    }  
23  
24    public void addLast(Pengunjung pengunjung) {  
25        Node newNode = new Node(pengunjung);  
26        if (head == null) {  
27            head = newNode;  
28            tail = newNode;  
29            head.next = head;  
30            head.prev = head;  
31        } else {  
32            tail.next = newNode;  
33            newNode.prev = tail;  
34            newNode.next = head;  
35            head.prev = newNode;  
36            tail = newNode;  
37        }  
38    }  
39}
```

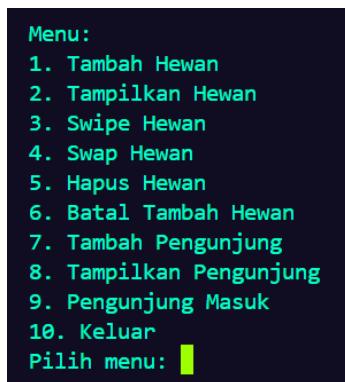
```

40     public Pengunjung removeFirst() {
41         if (head == null) {
42             return null;
43         }
44         Pengunjung removedData = head.data;
45         if (head == tail) {
46             head = null;
47             tail = null;
48         } else {
49             head = head.next;
50             head.prev = tail;
51             tail.next = head;
52         }
53         return removedData;
54     }
55
56     public boolean isEmpty() {
57         return head == null;
58     }
59
60     public boolean contains(String name) {
61         if (isEmpty()) {
62             return false;
63         }
64         Node current = head;
65         do {
66             if (current.data.getNama().equals(name)) {
67                 return true;
68             }
69             current = current.next;
70         } while (current != head);
71         return false;
72     }
73
74     public void display() {
75         if (isEmpty()) {
76             System.out.println("Tidak ada pengunjung dalam antrian.");
77             return;
78         }
79         Node current = head;
80         int i = 1;
81         do {
82             System.out.println(i + ". " + current.data.getNama());
83             current = current.next;
84             i++;
85         } while (current != head);
86     }
87 }
```

BAB IV

PETUNJUK PENGGUNAAN

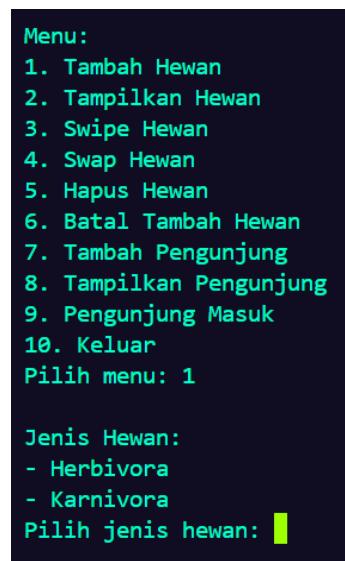
Pertama-tama, user yaitu seorang zookeeper akan tiba di menu page dimana sang zookeeper ini mampu melakukan beberapa fitur, seperti menambahkan hewan maupun pengunjung dan juga merubah posisi hewan, menghapus hewan, melihat posisi hewan serta memasukkan pengunjung ke dalam kebun binatang.



```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 1
```

Gambar 4.1 Menu awal

Kita akan mencoba Tambah Hewan, kita bisa melakukan ini dengan menginput angka urutannya yaitu 1 pada scanner.



```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 1  
  
Jenis Hewan:  
- Herbivora  
- Karnivora  
Pilih jenis hewan: 1
```

Gambar 4.2 Menu 1 Tambah Hewan

Setelah memasukkan 1, maka kita akan diberi opsi ingin memasukkan hewan karnivora atau herbivora. Silakan mengetikkan jenis hewan yang ingin ditambahkan.

Setelah menentukan jenis hewan, user akan diminta memasukkan informasi-informasi terkait hewan tersebut. Mulai dari nama, makanan, habitat, deskripsi, suara dan juga kegiatan. Jika semua input dari user sudah sesuai maka pesan berhasil akan ditampilkan.

```
Jenis Hewan:  
- Herbivora  
- Karnivora  
Pilih jenis hewan: Karnivora  
Nama hewan: Singa  
Makanan: Daging  
Habitat: Sabana  
Deskripsi: kucing berotot dengan tubuh panjang, kepala besar dan kaki pendek  
Suara: aummmmm  
Kegiatan: berburu  
Hewan berhasil ditambahkan!
```

Gambar 4.3 Berhasil Menambahkan Hewan

Setelah menambahkan sejumlah hewan, kita bisa kembali ke halaman utama dan memilih menu 2 untuk menampilkan hewan-hewan yang telah ditambahkan.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 2  
  
Daftar Hewan:  
1. Singa  
2. Sapi  
3. Beruang
```

Gambar 4.4 Menu 2 Tampilkan Hewan

Menu 3 berguna sebagai tempat kita dapat melihat daftar informasi dari setiap hewan. Pertama kali kita menggunakan menu 3 kita akan selalu tiba di INDEX 0 / hewan pertama. Namun kita bisa menggeser informasi ke hewan-hewan berikutnya maupun sebelumnya. Seperti yang bisa dilihat pada gambar di bawah ini, kita bisa memilih aksi 1 yaitu next ataupun aksi 2 yaitu previous. Satu detail penting soal ini adalah kita bisa melihat posisi kita

saat ini yaitu pada hewan yang sedang dilihat informasinya. Seperti pada gambar secara default kita akan berada di hewan pertama yaitu singa. Bila kita next (1) maka akan tiba di informasi sapi.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 3  
  
Informasi Hewan Saat Ini:  
Nama Hewan: Singa  
Jenis Makanan: Daging  
Habitat: Sabana  
Deskripsi dari Singa: kucing berotot dengan tubuh panjang, kepala besar dan kaki pendek  
Suara dari Singa: berburu  
Kegiatan dari Singa: aummmm  
  
Swipe Hewan:  
1. Swipe ke kanan (Next)  
2. Swipe ke kiri (Previous)  
3. Kembali ke Menu Utama  
Pilih aksi: 1
```

Gambar 4.5 Menu 3 Swipe Hewan

Bisa dilihat pada gambar dibawah ini setelah di next kita tiba di sapi, dan jika kita previous lagi maka akan kembali di singa. **Perlu diingat:** Fitur swipe ini akan terus mengingat kita berada di posisi mana dalam list. Jika kita merubah posisi dari hewan (swap) kita akan tetap berada di posisi yang sama jika kembali ke menu 3 swipe meskipun hewannya berubah.

```
Pilih aksi: 1  
  
Informasi Hewan:  
Nama Hewan: Sapi  
Jenis Makanan: rumput  
Habitat: padang rumput  
Deskripsi dari Sapi: hewan ternak yang memiliki 4 kaki dan bisa diternak untuk mendapatkan susu  
Suara dari Sapi: makan  
Kegiatan dari Sapi: mooo  
  
Swipe Hewan:  
1. Swipe ke kanan (Next)  
2. Swipe ke kiri (Previous)  
3. Kembali ke Menu Utama  
Pilih aksi: 1
```

Gambar 4.6 Menu 3 Swipe Hewan Next

```
Pilih aksi: 2

Informasi Hewan:
Nama Hewan: Singa
Jenis Makanan: Daging
Habitat: Sabana
Deskripsi dari Singa: kucing berotot dengan tubuh panjang, kepala besar dan kaki pendek
Suara dari Singa: berburu
Kegiatan dari Singa: aummm

Swipe Hewan:
1. Swipe ke kanan (Next)
2. Swipe ke kiri (Previous)
3. Kembali ke Menu Utama
Pilih aksi: 1
```

Gambar 4.7 Menu 3 Swipe Hewan Previous

Bila sudah selesai menggunakan menu swipe bisa memasukkan aksi 3 yaitu kembali ke menu utama. Seperti yang sudah kami singgung sebelumnya, kita bisa merubah posisi hewan menggunakan fitur Swap (4). Dari main menu langsung saja memasukkan menu 4, dan kita akan ditampilkan daftar hewannya kembali dengan urutan saat ini. Silahkan memasukkan urutan dari hewan-hewan yang ingin ditukar, sebagai contoh kita akan menukar hewan di posisi 1 dan 2 yaitu singa dan sapi. Jika input yang dimasukkan user sesuai permintaan (sesuai dengan urutan hewan) maka pesan berhasil akan muncul.

```
Menu:
1. Tambah Hewan
2. Tampilkan Hewan
3. Swipe Hewan
4. Swap Hewan
5. Hapus Hewan
6. Batal Tambah Hewan
7. Tambah Pengunjung
8. Tampilkan Pengunjung
9. Pengunjung Masuk
10. Keluar
Pilih menu: 4

Daftar Hewan:
1. Singa
2. Sapi
3. Beruang

Swap Hewan:
Masukkan urutan hewan pertama: 1
Masukkan urutan hewan kedua: 2
Hewan berhasil di-swap!
```

Gambar 4.8 Menu 4 Swap Hewan

Setelah berhasil menukar posisi hewan, kita akan dibawa kembali ke menu utama dan kita bisa menggunakan menu 2 kembali untuk menampilkan hewan dan bisa dilihat bahwa posisinya sudah tertukar antara 1 dan 2 dimana sapi dan singa telah bertukar posisi..

```
Daftar Hewan:  
1. Sapi  
2. Singa  
3. Beruang  
  
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 4
```

Gambar 4.9 Menu 2 Tampilkan Hewan sesudah Swap

Sekarang kita akan mencoba menu 5 yaitu delete hewan atau menghapus hewan. Untuk mencobanya, dari menu utama silahkan masukkan menu 5 delete hewan. Setelah itu kita akan ditampilkan daftar hewan saat ini, kemudian masukkanlah urutan dari hewan yang ingin dihapus.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 5  
  
Daftar Hewan:  
1. Sapi  
2. Singa  
3. Beruang  
  
Masukkan urutan hewan yang ingin dihapus: 2
```

Gambar 4.10 Menu 5 Delete Hewan

Setelah memasukkan urutan dari hewan yang ingin dihapus sesuai dengan permintaan (memasukkan urutan sesuai dengan hewan dalam list) maka pesan berhasil akan ditampilkan seperti pada gambar dibawah ini. Untuk memeriksa apakah hewannya sudah dihapus atau belum kita bisa menggunakan menu 2 kembali untuk menampilkan hewan saat ini. Bisa dilihat bahwa hewan 1 yaitu sapi telah kita hapus dan sudah tidak berada dalam daftar kita.

```
Daftar Hewan:  
1. Sapi  
2. Singa  
3. Beruang  
  
Masukkan urutan hewan yang ingin dihapus: 1  
Hewan berhasil dihapus!  
  
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 2  
  
Daftar Hewan:  
1. Singa  
2. Beruang
```

Gambar 4.11 Menu 2 Tampilkan Hewan setelah Delete

Sekarang kita akan mencoba fitur undo yaitu menu 6, fitur undo ini hanya berlaku untuk tambah hewan dan dapat digunakan apabila user salah menginput hewan. Untuk itu kita akan mencoba menambahkan hewan baru yaitu kambing.

```
Pilih menu: 1  
  
Jenis Hewan:  
- Herbivora  
- Karnivora  
Pilih jenis hewan: Herbivora  
Nama hewan: Kambing  
Makanan: rumput  
Habitat: padang rumput  
Deskripsi: berbulu lembut dan lebat  
Suara: mbee  
Kegiatan: terbang  
Hewan berhasil ditambahkan!  
  
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 1
```

Gambar 4.12 Menu 2 Tampilkan Hewan setelah Delete

Ternyata user salah menginput kegiatan kambing dan ingin melakukan undo atau pembatalan. Maka user bisa langsung memasukkan menu 6 batal tambah hewan, untuk langsung mengeluarkan binatang itu dari list. **Perlu diingat**, karena fitur ini memanfaatkan konsep stack maka undo atau batal hanya bisa dilakukan langsung setelah hewan tersebut ditambahkan. Karena jika sudah ditukar-tukar posisi hewannya maka sudah tidak selaras dengan concept stack yaitu LIFO atau Last in First Out. Jika ingin melakukan undo maka harus langsung setelah hewan ditambahkan.

```
Pilih menu: 2

Daftar Hewan:
1. Singa
2. Beruang
3. Kambing

Menu:
1. Tambah Hewan
2. Tampilkan Hewan
3. Swipe Hewan
4. Swap Hewan
5. Hapus Hewan
6. Batal Tambah Hewan
7. Tambah Pengunjung
8. Tampilkan Pengunjung
9. Pengunjung Masuk
10. Keluar
Pilih menu: 6
Aksi berhasil di-undo.
```

Gambar 4.13 Menu 6 Batal Tambah Hewan

Setelah kita memasukkan menu 6 batal tambah hewan, maka pesan berhasil akan muncul. Pada gambar dibawah ini kita melakukan cek kembali apakah hewannya sudah dibatalkan atau belum dan ternyata sudah menghilang dari list.

```
Menu:
1. Tambah Hewan
2. Tampilkan Hewan
3. Swipe Hewan
4. Swap Hewan
5. Hapus Hewan
6. Batal Tambah Hewan
7. Tambah Pengunjung
8. Tampilkan Pengunjung
9. Pengunjung Masuk
10. Keluar
Pilih menu: 2

Daftar Hewan:
1. Singa
2. Beruang
```

Gambar 4.14 Menu 2 Tampilkan Hewan setelah Undo

Sebagai zookeeper kita juga mampu untuk menambahkan pengunjung, oleh karena itu kita dapat menggunakan menu 7 untuk menambahkan pengunjung serta informasi-informasinya. Dengan ditambahnya pengunjung juga mereka akan masuk ke dalam daftar antrian kebun binatang.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 7  
Nama pengunjung: Tristan  
Umur pengunjung: 19  
Asal pengunjung: Jakarta  
Jenis kelamin pengunjung: Laki-laki  
Hewan favorit pengunjung: Singa  
Pengunjung berhasil ditambahkan ke antrian.  
  
Pengunjung saat ini sedang mengunjungi Kebun Binatang: Tristan  
Pengunjung Tristan sedang menunggu dalam antrian.
```

Gambar 4.15 Menu 7 Tambah Pengunjung

Seperti pada gambar dibawah ini, setiap pengunjung baru akan masuk ke dalam antrian dan menunggu sampai diberi aksi masuk untuk ke dalam kebun binatang.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 7  
Nama pengunjung: Karina  
Umur pengunjung: 19  
Asal pengunjung: Jakarta  
Jenis kelamin pengunjung: Perempuan  
Hewan favorit pengunjung: Beruang  
Pengunjung berhasil ditambahkan ke antrian.  
  
Pengunjung saat ini sedang mengunjungi Kebun Binatang: Tristan  
Pengunjung Karina sedang menunggu dalam antrian.
```

Gambar 4.16 Menu 7 Tambah Pengunjung

Setelah pengunjung ditambahkan kita bisa melihat keadaan antriannya memiliki berapa pengunjung dan nama-namanya.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 8  
  
Antrian Pengunjung:  
1. Tristan  
2. Karina  
3. Jonathan
```

Gambar 4.17 Menu 8 Tampilkan Pengunjung

Terakhir kita akan memberikan aksi masuk untuk pengunjung satu per satu menggunakan menu 9 pengunjung masuk yang menggunakan konsep queue yaitu FIFO atau First in First out.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 9  
  
Pengunjung Tristan sekarang sedang mengunjungi Kebun Binatang.
```

Gambar 4.18 Menu 9 Pengunjung Masuk

Satu per satu, pengunjung akan memasuki kebun binatang setiap menu 9 digunakan dan di saat yang bersamaan, antrian pengunjung akan berkurang seiring antrian paling depan masuk. Bisa dilihat dari screenshot-screenshot dibawah ini.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 8  
  
Antrian Pengunjung:  
1. Karina  
2. Jonathan
```

Gambar 4.19 Menu 8 Tampilkan Pengujung setelah Masuk

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 9  
  
Pengunjung Karina sekarang sedang mengunjungi Kebun Binatang.
```

Gambar 4.20 Menu 9 Pengunjung Masuk

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 8  
  
Antrian Pengunjung:  
1. Jonathan
```

Gambar 4.21 Menu 8 Tampilkan Pengunjung setelah Masuk

Terakhir menu 10 keluar dapat kita gunakan untuk keluar dari program. Pesan terima kasih akan muncul sebagai bentuk apresiasi telah mengunjungi kebun binatang kami.

```
Menu:  
1. Tambah Hewan  
2. Tampilkan Hewan  
3. Swipe Hewan  
4. Swap Hewan  
5. Hapus Hewan  
6. Batal Tambah Hewan  
7. Tambah Pengunjung  
8. Tampilkan Pengunjung  
9. Pengunjung Masuk  
10. Keluar  
Pilih menu: 10  
Terima kasih sudah berkunjung ke kebun binatang.  
PS C:\Users\Karina\eclipse-workspace\Kebun Binatang\Kebun-Binatang> █
```

Gambar 4.22 Menu 10 Keluar

BAB V

KESIMPULAN

1. Penjelasan Kebun Binatang

Code kami adalah mengangkat tema Kebun Binatang, dimana kebun binatang ini mengimplementasikan seluruh modul yang ada pada Object Oriented Program. Kebun Binatang dapat memilih menu untuk memasukan daftar hewan, deskripsi hewan, melihat kegiatan hewan, dan juga membuat list pengunjung. Modul yang kami gunakan adalah sebagai berikut:

- **Encapsulation:** Terdapat ciri khas pada code yaitu GETTER & SETTER. Pada kebun binatang kami gunakan untuk mengambil nama di class Pengunjung dan Hewan.
 - **Kelas Hewan Halaman 4 (Line 20-54)**
 - **Kelas Pengunjung Halaman 6 (Line 18-56)**
- **Inheritance:** Terdapat ciri khas pada code yaitu kata kunci “extends” digunakan untuk menunjukkan bahwa sebuah kelas adalah subclass dari kelas lain, dan anotasi “@Override” sering digunakan untuk menunjukkan bahwa metode dalam subclass menimpa metode dari superclass. Pada kebun binatang kami gunakan untuk menurunkan class Hewan ke Karnivora dan Dari kelas Karnivora ke Herbivora.
 - **Kelas Hewan Halaman 9 (Line 3 - 84)**
 - **Kelas Karnivora Halaman 10 (Line 3 - 27)**
 - **Kelas Herbivora Halaman 10 (Line 3 - 8)**
- **Interface:** Sebuah kelas yang mengimplementasikan interface harus menyediakan implementasi untuk semua metode yang

dideklarasikan dalam interface. Kelas tersebut menggunakan kata kunci implements dan interface pada class ZooInterface.

- **Kelas ZooInterface Halaman 16 (Line 1 - 15)**

- **Abstraction:** Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan mungkin berisi metode yang tidak memiliki implementasi (metode abstrak). Kelas abstrak fokus pada apa yang dilakukan suatu objek, bukan bagaimana objek tersebut melakukannya. Detail implementasi tersembunyi di balik antarmuka atau kelas abstrak.

- **Kelas Hewan Halaman 9 (Line 3 - 9)**

- **Polymorphism:** Memungkinkan metode yang sama untuk dipanggil pada objek yang berbeda, yang menghasilkan perilaku yang berbeda tergantung pada jenis objek yang memanggil metode tersebut.

- **Kelas Herbivora Halaman 10 (Line 3 - 8)**

- **ArrayList:** Array Hewan[] hewanArray digunakan untuk menyimpan objek Hewan. Konsep dasar array dan ArrayList adalah sama dalam hal menyimpan koleksi objek.

- **Kelas ZooKeeper Halaman 20 (Line 23-24)**
- **Kelas ZooKeeper Halaman 21 (Line 44-45)**
- **Kelas ZooKeeper Halaman 18 (Line 9-37)**

- **Single Linked List:** VisitorQueue class: Menggunakan implementasi linked list untuk mengantri pengunjung. Class Node digunakan untuk merepresentasikan node dalam linked list, dengan

atribut data (yang menyimpan objek Pengunjung) dan next (referensi ke node berikutnya).

- **Kelas VisitorQueue (method tampilkanPengunjung)**

Halaman 14 (Line 1 - 15)

- **Double Linked List:** Double Linked List (DLL) adalah sebuah struktur data linear yang terdiri dari node-node di mana setiap node memiliki dua pointer, satu menunjuk ke node berikutnya (next) dan satu lagi menunjuk ke node sebelumnya (prev).

- **Kelas Zookeeper Halaman 13 (Line 1 -46)**

- **Circular Linked List:** Circular Linked List (CLL) adalah sebuah struktur data linked list di mana node terakhir menunjuk kembali ke node pertama, membentuk suatu lingkaran.

- **Kelas CircularList Halaman 18 (Line 3 - 46),**

- **Kelas CircularList Halaman 19 (Line 1 -37)**

- **Double Circular Linked List:** Double Circular Linked List (DCLL) adalah kombinasi dari double linked list dan circular linked list di mana setiap node memiliki dua pointer (next dan prev) dan node terakhir menunjuk kembali ke node pertama, serta node pertama menunjuk kembali ke node terakhir.

- **Kelas DoubleCircularLinkedList Halaman 24-25 (Line 3-87)**

- **Stack:** ZooKeeper class: Menggunakan stack untuk undo actions. UndoAction[] undoStack digunakan untuk menyimpan tindakan

yang bisa di-undo, dengan operasi push dan pop diimplementasikan.

- **Kelas UndoAction Halaman 12 (Line 3-39)**

- **Queue:** VisitorQueue class: Menggunakan metode enqueue dan dequeue untuk mengantri dan mengeluarkan pengunjung dari antrian, yang merupakan karakteristik dari queue.
 - **Kelas VisitorQueue Halaman 13 (Line 3 - 42) ,**
 - **Kelas VisitorQueue Halaman 14 (Line 1 - 38),**
 - **Kelas VisitorQueue Halaman 15 (Line 1- 35)**

DAFTAR PUSTAKA

- [1] **Niemeyer, Patrick, and Daniel Leuck.** Learning Java. O'Reilly Media, 2021.
- [2] **Desmoulins, Nadège, et al.** Zoo Animal Learning and Training: An Introduction to Husbandry Training. Springer, 2019.