UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e de Engenharia

2022/2023

Programação Orientada por Objetos

"Jumanji"

Grupo 29

Docentes:

Luís Ferreira
Bernardo Gouveia
Sergi Bermúdez

Introdução

Este relatório tem por objetivo demonstrar e explicar o funcionamento de uma aplicação que permite gerir um zoo, usando para tal a plataforma *NetBeans IDE* e a linguagem de programação *Java*.

Este relatório explicará os objetivos principais deste trabalho, tal como a forma de implementação e os procedimentos realizados para tal fim e a justificação pela qual decidiu-se implementá-los. E no final, é incluído, em anexo, o diagrama UML, que demonstra toda a estruturação do software desenvolvido.

Este trabalho tem por objetivos, aplicar os conhecimentos adquiridos na unidade curricular de Programação Orientada por Objetos com o propósito de criar uma aplicação que permita simular um zoo tal como inserir e consultar a informação que pretende.

Resumidamente serão dadas ao utilizador várias opções, entre as quais, adquirir animais, estes têm as suas próprias características tal como o seu respetivo preço.

O utilizador poderá criar instalações para alojar os animais, este poderá também ver o obituário e o histórico com todas as ações já realizadas na aplicação.

Sempre que o utilizador desejar poderá também selecionar a opção "período contabilístico" que representará um ano, com o passar dos "anos", os animais poderão morrer e se reproduzir, sendo que o utilizador terá de ter atenção á situação económica do zoo.

Procedimento e implementação do código

Classes

1. Animal (Abstrata)

Os principais atributos são o atributo int "age" (que determina a sua idade), o atributo double "attractiveness" (quantifica o nível de atratividade que este animal tem) e o atributo int "price" (diz quanto vale este animal).

Além disso também contém os arrays que guardam os nomes dos animais, as espécies dos mesmos, os animais do ano chinês, as características, as mutações e métodos selectores e modificadores.

1.1(Todos os Animais)

As classes Boi, Cabra, Cao, Cavalo, Chita, Cobra, Coelho, Dragao, Galo Jaguar, Leao, Macaco, Porco, Rato, Tigre e Zebra são todas subclasses da classe Animal.

Nos construtores de cada uma delas, chamamos a função **setTimeLife** e definimos que o animal vive entre x a y anos, chamamos também a função **setMaintenance** e definimos um determinado valor para cada animal.
As classes Cavalo e Zebra implementam a interface Equus. As classes Tigre, Leao e Jaguar implementam a interface Panthera que lhes permite rugir.

2. Installation

Possui os atributos int “capacity” (guarda a capacidade da instalação), int price (guarda o preço a instalação) e um array ArrayList<Animal> “animals” (guarda os animais que estão nessa instalação). Além de métodos seletores também tem um método para adicionar novos animais à instalação(public void addAnimals(Animal oneAnimal)), outro para verificar se a instalação está cheia() e um override no método "toString()" para mostrar o id da instalação, o preço, a capacidade e quantos animais estão lá presentes.

3. Main

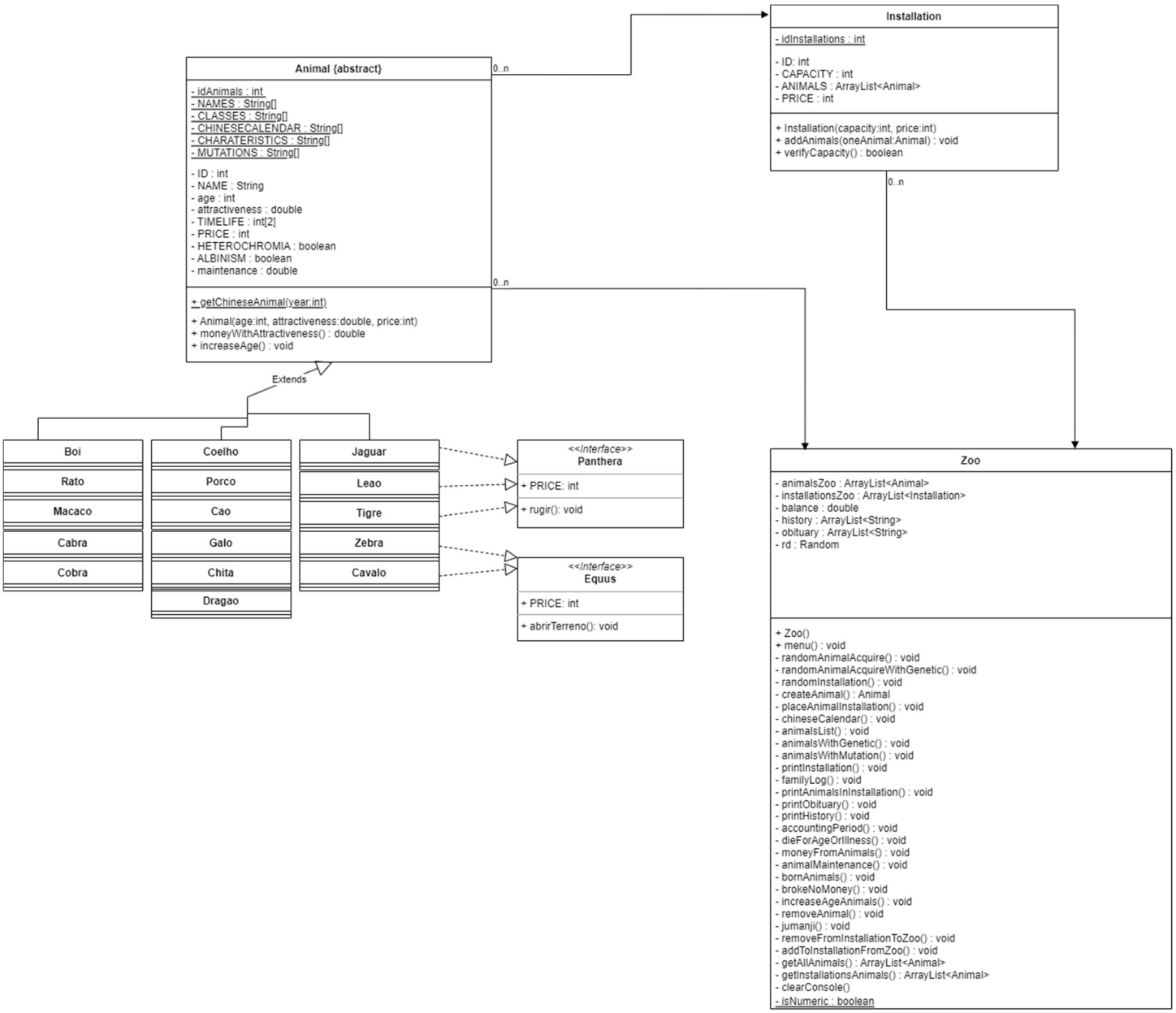
Inicía o zoo e o menu do mesmo.

4. Zoo

A classe Zoo contém o array que guarda todos os animais presentes no zoo, dentro ou não de um instalação (private final ArrayList<Animal> animalsZoo), o array que guarda as instalações do zoo (private final ArrayList<Installation>installationsZoo), o balanço económico do zoo (private double balance), o array que guarda o histórico (ArrayList<Installation> installationsZoo) e o array que guarda o obituário (private final ArrayList<String> obituary).

Possui o menu usado no programa e os métodos necessários para cada opção do menu.

DIAGRAMA UML



Código

Classe Main:

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Main {

    public static void main(String[] args) {
        Zoo myZoo = new Zoo();
        myZoo.menu();
    }
}
```

Classe Zoo

```
package com.mycompany.zoo;

import java.util.Scanner;
import java.util.ArrayList;
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import java.util.Random;

public final class Zoo {

    private final ArrayList<Animal> animalsZoo; //Array with Zoo
animals
    private final ArrayList<Installation> installationsZoo; //Array
with Zoo installations
    private double balance; //Balance of the User - Starts with 20000
    private final ArrayList<String> history; //Array with the Zoo
history
    private final ArrayList<String> obituary; //Array with the Zoo
obituary
    private final Random rd; //Random variable for the all the program

    public Zoo() {
        animalsZoo = new ArrayList<>();
        installationsZoo = new ArrayList<>();
        history = new ArrayList<>();
        obituary = new ArrayList<>();
        balance = 20000;
        rd = new Random();
    }

    //Zoo menu
    public void menu() {

        boolean leave = false;
```

```
        clearConsole();
        Scanner option = new Scanner(System.in, "cp1252");

        //Option verification for the Menu
        do {

            System.out.println("Saldo atual: " + balance);
            display();

            switch (option.next()) {
                case "1" -> {
                    randomAnimalAcquire();
                }
                case "2" -> {
                    randomAnimalAcquireWithGenetic();
                }
                case "3" -> {
                    randomInstallation();
                }
k
                    placeAnimalInstallation();
                }
                case "5" -> {
                    chineseCalendar();
                }
                case "6" -> {
                    animalsList();
                }
                case "7" -> {
                    animalsWithGenetic();
                }
                case "8" -> {
                    animalsWithMutation();
                }
                case "9" -> {
                    printInstallation();
                }
                case "10" -> {
                    familyLog();
                }
                case "11" -> {
                    printObituary();
                }
                case "12" -> {
                    printHistory();
                }
                case "13" -> {
                    accountingPeriod();
                }
                case "14" -> {
                    jumanji();
                }
                case "0" -> {
                    leave = true;
                }
                default -> System.out.println("Selecione uma das
opções!");
```

```
            }
        } while (!leave);

    }

    private void display() {

        System.out.println("(1)Adquirir animal aleatório");
        System.out.println("(2)Adquirir animal com caraterística
genética");
        System.out.println("(3)Construir instalação");
        System.out.println("(4)Colocar animal em instalação");
        System.out.println("(5)Calendário chinês");
        System.out.println("(6)Listar animais");
        System.out.println("(7)Listar animais com dada caraterística
genética");
        System.out.println("(8)Listar animais com dada mutação");
        System.out.println("(9)Listar instalações");
        System.out.println("(10)Retrato de família animal");
        System.out.println("(11)Obituário");
        System.out.println("(12)Histórico");
        System.out.println("(13)Período contabilístico");
        System.out.println("(14)Jumanji");
        System.out.println("(0)Sair da aplicação");

    }

    //First Menu option
    private void randomAnimalAcquire() {


        System.out.println("Escolha um dos seguintes animais:");
        Animal[] option = new Animal[3];
        //Creation of 3 random animals
        for (int i = 0; i < 3; i++) {
            int idxRandom = rd.nextInt(Animal.getClasses().length);
            System.out.print((i+1)+")");
            option[i] = createAnimal(Animal.getClasses()[idxRandom],
rd.nextInt(10));
            System.out.println(option[i].toString());
        }
        boolean leave = false;
        Scanner validation = new Scanner(System.in, "cp1252");

        //User's choice verification
        do {

            String oneoption = validation.next();
            switch (oneoption) {
                case "1", "2", "3" -> {
                    int myOption = Integer.parseInt(oneoption);
                    Animal optionAnimal = option[myOption-1]; //Chosen
animal
                    payWithMoney(optionAnimal.getPRICE()); //Pay for
the chosen animal
```

```
                    animalsZoo.add(optionAnimal); //Add the animal to
the ZOO
                    history.add("Adquiriu: " +
optionAnimal.toString());
                    leave = true;
                }
                case "0" -> {
                    leave = true;
                }
                default -> System.out.println("Insira um número
válido");
            }

        } while (!leave);



    }

    //Second Menu option
    private void randomAnimalAcquireWithGenetic() {

        boolean leave = false;
        String oneoption;
        Scanner validation = new Scanner(System.in, "cp1252");

        //Print all animal charateristics
        System.out.println("Escolha uma das seguintes
caraterísticas:");
        for (String oneCharateristic : Animal.getCharateristics())
            System.out.println("- " + oneCharateristic);

        //User's choice verification
        do {

            oneoption = validation.next();
            for (String oneCharateristic : Animal.getCharateristics())
                if (oneoption.equals(oneCharateristic)) leave = true;

            if (!leave)
                System.out.println("Selecione uma das opções!");
        } while (!leave);

        String[] classes = Animal.getClasses(); // Get a list of all
classes

        Object interfaceClass = null;
        try {
            interfaceClass =
Class.forName("com.mycompany.zoo."+oneoption); //Get the interface
class for the chosen charateristic
        } catch (ClassNotFoundException ex) {
            System.out.println("Caraterística não encontrada!");
        }
        ArrayList<String> classesWithChar = new ArrayList<>();
        for (String cls : classes) {
            try {
```

```
                if
(((Class)interfaceClass).isAssignableFrom(Class.forName("com.mycompany
.zoo."+cls)))
                    classesWithChar.add(cls); // All classes with the
chosen charateristics

            } catch (ClassNotFoundException ex) {
                System.out.println("Animal não encontrado!");
            }
        }

        Animal oneAnimal;
        oneAnimal =
createAnimal(classesWithChar.get(rd.nextInt(classesWithChar.size())),
rd.nextInt(10));

        animalsZoo.add(oneAnimal);
        payWithMoney(oneAnimal.getPRICE());
        System.out.println("Adquiriu: " + oneAnimal.toString());
        history.add("Adquiriu: " + oneAnimal.toString() + " com a
genética " + oneoption);
    }

    //Third Menu option
    private void randomInstallation() {

        System.out.println("Escolha uma das seguintes instalações:");
        int[][] option = new int[3][2];

        //Print all installations
        for (int i = 0; i < 3; i++) {
            System.out.print((i+1)+")");
            option[i][0] = rd.nextInt(2,10);
            option[i][1] = rd.nextInt(1000, 10000);
            System.out.println("Capacidade: " + option[i][0] + " |
Price: " + option[i][1]);
        }
        System.out.println("0) Sair");

        boolean leave = false;
        Scanner validation = new Scanner(System.in, "cp1252");

        //User's choice verification
        do {

            String oneoption = validation.next();
            switch (oneoption) {
                case "1", "2", "3" -> {
                    int myOption = Integer.parseInt(oneoption);
                    leave = true;
                    balance -= option[myOption-1][1];
                    Installation oneInstallation = new
Installation(option[myOption-1][0], option[myOption-1][1]);
                    installationsZoo.add(oneInstallation);
                    history.add("Instalação " +
oneInstallation.getID() + " criada.");
                }
```

```
                case "0" -> leave = true;
                default -> System.out.println("Insira um número
válido!");
            }

        } while (!leave);

    }

    //Creation of a new Animal depending on the chosen Animal
    private Animal createAnimal(String oneClass, int oneAge) {

        Animal oneAnimal;
        double oneAttractiveness = rd.nextDouble(10);
        int onePrice = rd.nextInt(100,1000);

        switch (oneClass) {
            case "Boi" -> oneAnimal = new Boi(oneAge,
oneAttractiveness, onePrice);
            case "Cabra" -> oneAnimal = new Cabra(oneAge,
oneAttractiveness, onePrice);
            case "Cao" -> oneAnimal = new Cao(oneAge,
oneAttractiveness, onePrice);
            case "Cavalo" -> oneAnimal = new Cavalo(oneAge,
oneAttractiveness, onePrice);
            case "Chita" -> oneAnimal = new Chita(oneAge,
oneAttractiveness, onePrice);
            case "Cobra" -> oneAnimal = new Cobra(oneAge,
oneAttractiveness, onePrice);
            case "Coelho" -> oneAnimal = new Coelho(oneAge,
oneAttractiveness, onePrice);
            case "Dragao" -> oneAnimal = new Dragao(oneAge,
oneAttractiveness, onePrice);
            case "Galo" -> oneAnimal = new Galo(oneAge,
oneAttractiveness, onePrice);
            case "Jaguar" -> oneAnimal = new Jaguar(oneAge,
oneAttractiveness, onePrice);
            case "Leao" -> oneAnimal = new Leao(oneAge,
oneAttractiveness, onePrice);
            case "Macaco" -> oneAnimal = new Macaco(oneAge,
oneAttractiveness, onePrice);
            case "Porco" -> oneAnimal = new Porco(oneAge,
oneAttractiveness, onePrice);
            case "Rato" -> oneAnimal = new Rato(oneAge,
oneAttractiveness, onePrice);
            case "Tigre" -> oneAnimal = new Tigre(oneAge,
oneAttractiveness, onePrice);
            case "Zebra" -> oneAnimal = new Zebra(oneAge,
oneAttractiveness, onePrice);
            default -> oneAnimal = new Boi(oneAge, oneAttractiveness,
onePrice);
        }
        return oneAnimal;
    }

    //Fourth Menu option
    private void placeAnimalInstallation() {
```

```
        if (installationsZoo.isEmpty()) {System.out.println("Não
existem instalações!"); return; }
        System.out.println("Escolha uma das seguintes instalações:");

        //Print Installations
        for (Installation oneInstallation : installationsZoo) {
            if (oneInstallation.verifyCapacity())
                System.out.println("- " + oneInstallation.getID() +
"(FULL)");
            else
                System.out.println("- " + oneInstallation.getID());
        }


        boolean leave = false;
        int oneoption;
        Installation optionedInstallation = null;
        Scanner validation = new Scanner(System.in, "cp1252");

        //User's choice verification
        do {

            String myOption = validation.next();
            if (isNumeric(myOption)) {
                oneoption = Integer.parseInt(myOption);

                for (Installation oneInstallation : installationsZoo)
                    if (oneInstallation.getID() == oneoption) {
                        leave = true;
                        optionedInstallation = oneInstallation;
                        break;
                    }

                if (!leave)
                    System.out.println("Insira um ID válido da
instalação!");
            } else
                System.out.println("Insira um ID válido da
instalação!");


        } while (!leave);

        //Print the list of animals in the Zoo
        System.out.println("Escolha um dos seguintes animais para
colocar na instalação:");
        for (Animal oneAnimal : animalsZoo)
            System.out.println("- " + oneAnimal.getId());


        leave = false;
        Animal optionedAnimal = null;

        //User's choice verification for the animal
        do {
```

```
            String myOption = validation.next();
            if (isNumeric(myOption)) {
                oneoption = Integer.parseInt(myOption);

                for (Animal oneAnimal : animalsZoo)
                    if (oneAnimal.getId() == oneoption) {
                        leave = true;
                        optionedAnimal = oneAnimal;
                        break;
                    }

                if (!leave)
                    System.out.println("Insira um ID válido do
Animal!");
            } else
                System.out.println("Insira um ID válido do Animal!");


        } while (!leave);

        addToInstallationFromZoo(optionedInstallation,
optionedAnimal);

    }

    //Fifth Menu option
    private void chineseCalendar() {

        boolean leave = false;
        int oneoption = 0;
        Scanner validation = new Scanner(System.in, "cp1252");

        System.out.print("Insira o ano: ");

        do {

            String myOption = validation.next();

            if (isNumeric(myOption)) {
                oneoption = Integer.parseInt(myOption);
                leave = true;
            } else
                System.out.println("Insira um número!");


        } while (!leave);

        for (Animal oneAnimal : getAllAnimals())
            if
(Animal.getChineseAnimal(oneoption).equals(oneAnimal.getClass().getSim
pleName()))

oneAnimal.setAttractiveness(oneAnimal.getAttractiveness() + (10-
oneAnimal.getAttractiveness()) * 0.5); //Increase the attractiveness
(max: 10)
```

```
        System.out.println("O animal do ano é: " +
Animal.getChineseAnimal(oneoption) + " - Boost atribuído!");
    }

    //Sixth Menu option
    private void animalsList() {

        for (Animal oneAnimal : getAllAnimals())
            System.out.println(oneAnimal.toString());

    }

    //Seventh Menu option
    private void animalsWithGenetic(){

       boolean leave = false;
       String option;
       Object interfaceClass = null;
       Scanner scan = new Scanner(System.in,"cp1252");

       //Print all animal charateristics
       System.out.println("Escolha uma característica");
       for (String oneCharateristic : Animal.getCharateristics())
                System.out.println("- " + oneCharateristic);

       //User's choice verification
       do {

            option = scan.next();
            for (String oneCharateristic : Animal.getCharateristics())
                if (option.equals(oneCharateristic)){
                    leave = true;
                    break;
                }
            if (!leave)
                System.out.println("Selecione uma das opções!");

        } while (!leave);
        try {
            interfaceClass =
Class.forName("com.mycompany.zoo."+option);
        } catch (ClassNotFoundException ex) {
            System.out.println("Caraterística não encontrada!");
        }
        try {

            for(Animal oneAnimal : getAllAnimals())

if(((Class)interfaceClass).isAssignableFrom(Class.forName(oneAnimal.ge
tClass().getName())))
                    System.out.println(oneAnimal.toString());

        }
        catch (ClassNotFoundException ex) {
                System.out.println("Animal não encontrado!");
        }
    }
```

```java
//Eighth Menu option
private void animalsWithMutation(){

    boolean leave = false;
    String option;
    Scanner scan = new Scanner(System.in,"cp1252");

    System.out.println("Escolha uma característica");
    for (String oneMutation : Animal.getMutations())
            System.out.println("- " + oneMutation);
    do {

        option = scan.next();
        for (String oneMutation : Animal.getMutations())
            if (option.equals(oneMutation))
                leave = true;

        if (!leave)
            System.out.println("Selecione uma das opções!");

    } while (!leave);

    switch(option){
        case "heterochromia" -> {
            for(Animal oneAnimal : getAllAnimals())
                if(oneAnimal.getHeterochromia())
                   System.out.println(oneAnimal.toString());

        }
        case "albinism" -> {
            for(Animal oneAnimal : getAllAnimals())
                if(oneAnimal.getAlbinism())
                   System.out.println(oneAnimal.toString());


        }
    }
}

//Ninth Menu Option
private void printInstallation() {

    for (Installation oneInstallation : installationsZoo)
        System.out.println(oneInstallation.toString());

}

//Tenth Menu Option
private void familyLog(){

    System.out.println("--RETRATO DE FAMILIA ANIMAL--");

    animalsList();
    printAnimalsInInstallation();
    printObituary();
    printHistory();
```

```
    }

    //Auxiliar function to the 10th option
    private void printAnimalsInInstallation(){

        for (Installation oneInstallation : installationsZoo){
            System.out.println(oneInstallation.toString());
            for (Animal oneAnimal : oneInstallation.getANIMALS())
                System.out.println("- " + oneAnimal.toString());
        }

    }

    //Eleventh Menu option
    private void printObituary() {

        System.out.println("--Obituário--");
        for (String oneString : obituary)
            System.out.println(oneString);

    }

    //Twelfth Menu option
    private void printHistory() {

        System.out.println("--História--");
        for (String oneString : history)
            System.out.println(oneString);

    }

    //Thirteenth Menu option
    private void accountingPeriod() {

        moneyFromAnimals();
        animalMaintenance();

        dieForAgeOrIllness();
        bornAnimals();
        brokeNoMoney();

        increaseAgeAnimals();

    }

    //Determine animals that die for age or illness - Accounting
period
    private void dieForAgeOrIllness() {

        ArrayList<Animal> animalsToRemove = new ArrayList<>();

        //Determine if an animal dies for eld
        for (Animal oneAnimal : getAllAnimals())
            if ((oneAnimal.getAge() >= oneAnimal.getTIMELIFE()[0] &&
rd.nextInt(100) < 10) || oneAnimal.getAge() >=
oneAnimal.getTIMELIFE()[1] ) {
```

```
                history.add("- O animal " + oneAnimal.getId() + "
morreu de velhice.");
                obituary.add(oneAnimal.toString());
                animalsToRemove.add(oneAnimal);
            }
        //Remove the animal from the whole Zoo
        for (Animal oneAnimal : animalsToRemove)
            removeAnimal(oneAnimal);

        animalsToRemove = new ArrayList<>();

        //Determine if an animal dies for illness
        for (Animal oneAnimal : getAllAnimals())
            if (rd.nextInt(100) < 2) {
                history.add("- O animal " + oneAnimal.getId() + "
morreu de doença.");
                obituary.add(oneAnimal.toString());
                animalsToRemove.add(oneAnimal);
            }

        //Remove the animal from the whole Zoo
        for (Animal oneAnimal : animalsToRemove)
            removeAnimal(oneAnimal);

    }

    //Money received from the animals in the installations -
Accounting period
    private void moneyFromAnimals() {

        for (Animal oneAnimal : getInstallationsAnimals())
            receiveMoney(oneAnimal.moneyWithAttractiveness());

    }

    //Money taken from the animal maintenance - Accounting period
    private void animalMaintenance() {

        for (Animal oneAnimal : getAllAnimals())
            payWithMoney(oneAnimal.getMaintenance());

    }

    //New born Animals on the Zoo - Accounting period
    private void bornAnimals() {

        ArrayList<Animal> animals = new ArrayList<>();

        for (Animal oneAnimal : getAllAnimals())
            if (rd.nextInt(100) < 5) {
                Animal myAnimal =
createAnimal(oneAnimal.getClass().getSimpleName(), 0);
                animals.add(myAnimal);
                history.add("+ Nascimento de um " +
oneAnimal.getClass().getSimpleName() + " " + myAnimal.getId());
            }
        animalsZoo.addAll(animals);
```

```
    }

    //Animals that run away from the Zoo with no maintenance or a
minor chance to leave - Accounting period
    private void brokeNoMoney() {

        ArrayList<Animal> animalsToRemove = new ArrayList<>();

        for (Animal oneAnimal : getAllAnimals())
            if (rd.nextInt(100) < 10 && balance <= 0)
                animalsToRemove.add(oneAnimal);

        for (Animal oneAnimal : animalsToRemove) {
            history.add("Animal " + oneAnimal.getId() + " fugiu!");
            removeAnimal(oneAnimal);
        }

    }

    //Function to increase the age of all animals - Accounting period
    private void increaseAgeAnimals() {

        for (Animal oneAnimal : getAllAnimals())
            oneAnimal.increaseAge();


    }

    //Remove an animal from all the ZOO
    private void removeAnimal(Animal myAnimal) {

        animalsZoo.remove(myAnimal);

        for (Installation oneInstallation : installationsZoo) {
            oneInstallation.getANIMALS().remove(myAnimal);
            break;
        }

    }

    //Fourteenth Menu Option
    private void jumanji() {

        ArrayList<Animal> animalsToRemove = new ArrayList<>();
        ArrayList<Animal> animalsToMoveToZoo = new ArrayList<>();

        //Determine if an animal leave the Zoo
        for (Animal oneAnimal : animalsZoo)
            if (rd.nextInt(100) < 10)
                animalsToRemove.add(oneAnimal);


        for (Animal oneAnimal : getInstallationsAnimals())
            //Determine if an animal leaves the installation to the
Zoo
            if (rd.nextInt(100) < 33)
```

```
                animalsToMoveToZoo.add(oneAnimal);
            //Determine if an animal leaves the installation and the
whole Zoo
            else if (rd.nextInt(100) > 67)
                animalsToRemove.add(oneAnimal);

        //Remove the animals from the whole Zoo
        for (Animal oneAnimal : animalsToRemove)
            removeAnimal(oneAnimal);

        //Move the animals to the Zoo
        for (Animal oneAnimal : animalsToMoveToZoo)
            removeFromInstallationToZoo(oneAnimal);

    }

    //Remove an animal from installation and add it to the Zoo
    private void removeFromInstallationToZoo(Animal oneAnimal) {

        animalsZoo.add(oneAnimal);
        for (Installation oneInstallation : installationsZoo)
            oneInstallation.getANIMALS().remove(oneAnimal);

        history.add("Animal " + oneAnimal.getId() + " saiu da
instalação!");

    }

    //Remove an animal from the Zoo and add it to the Installation
    private void addToInstallationFromZoo(Installation
oneInstallation, Animal oneAnimal) {

        if (!oneInstallation.verifyCapacity())
            oneInstallation.getANIMALS().add(oneAnimal);
        else {
            int idx = rd.nextInt(oneInstallation.getANIMALS().size());
            Animal theAnimal = oneInstallation.getANIMALS().get(idx);
            removeFromInstallationToZoo(theAnimal);
            oneInstallation.getANIMALS().add(idx, oneAnimal);
        }
        animalsZoo.remove(oneAnimal);
        history.add("Animal " + oneAnimal.getId() + " entrou na
instalação " + oneInstallation.getID());

    }

    //Verify if a string is numeric
    private static boolean isNumeric(String strNum) {

        if (strNum == null) {
            return false;
        }
        try {
            double d = Double.parseDouble(strNum);
        } catch (NumberFormatException nfe) {
            return false;
        }
```

```
        return true;

    }

    private void payWithMoney(double pay) {
        balance -= pay;
    }

    private void receiveMoney(double money) {
        balance += money;
    }

    //Auxiliar function to get all animals in the whole Zoo
    private ArrayList<Animal> getAllAnimals() {

        ArrayList<Animal> allAnimals = new ArrayList<>();

        allAnimals.addAll(animalsZoo);
        allAnimals.addAll(getInstallationsAnimals());

        return allAnimals;

    }

    //Auxiliar function to get all animals inside the Installations
    private ArrayList<Animal> getInstallationsAnimals() {

        ArrayList<Animal> installationAnimals = new ArrayList<>();

        for (Installation oneInstallation : installationsZoo) {
            installationAnimals.addAll(oneInstallation.getANIMALS());
        }

        return installationAnimals;

    }

    //Function to clear the Netbeans console
    private void clearConsole() {

        try {
            Robot limpa = new Robot();
            limpa.keyPress(KeyEvent.VK_CONTROL);
            limpa.keyPress(KeyEvent.VK_L);
            limpa.keyRelease(KeyEvent.VK_CONTROL);
            limpa.keyRelease(KeyEvent.VK_L);
            try {
                Thread.sleep(10);
            } catch (InterruptedException ex) {
                System.out.println("ERRO");
            }

        } catch (AWTException e) {
            System.out.print("ERRO\n");
        }
        System.out.println("--------------------------------Jumanji---
-------------------------------");
```

```
    }
}
```

Classe Animal

```
package com.mycompany.zoo;

import java.util.Objects;
import java.util.Random;

public abstract class Animal {

    private static int idAnimals = 0;
    private static final String[] NAMES = {"Abelha Maia", "Salvador",
"BBC", "George Floyd", "Maiq Taison", "Buba J", "Amanda D. P. Throat",
"Ligma", "Anita Hardcok", "Banana Hammock", "Ben Dover", "Betty
Humper", "Dick Long", "Dixie Rect", "E. Normous Peter", "Fluffy
Cookie", "Harry Cox", "Harry Maguire", "Marion Money", "Stella
Virgin", "Yuri Nator", "Tiger Style", "Captain Morgan", "Edward
Cocaine", "Bouncy Nuggets", "Willie Stroker", "Emerson Bigguns",
"Annie Rection", "Bob Maddick", "Mike Hawk", "CR7 SEWEYYYY"};
    private static final String[] CLASSES = {"Boi", "Cabra", "Cao",
"Cavalo", "Chita", "Cobra", "Coelho", "Dragao", "Galo", "Jaguar",
"Leao", "Macaco", "Porco", "Rato", "Tigre", "Zebra"};
    private static final String[] CHINESECALENDAR = {"Macaco", "Galo",
"Cao", "Porco", "Rato", "Boi", "Tigre", "Coelho", "Dragao", "Cobra",
"Cavalo", "Cabra"};
    private static final String[] CHARATERISTICS = {"Equus",
"Panthera"};
    private static final String[] MUTATIONS = {"heterochromia",
"albinism"};
    private final int ID;
    private final String NAME;
    private int age;
    private double attractiveness;
    private final int[] TIMELIFE = new int[2];
    private final int PRICE;
    private final boolean HETEROCHROMIA;
    private final boolean ALBINISM;
    private double maintenance = 0; //Maintenance cost for each animal

    public Animal(int age, double attractiveness, int price) {

        Random rd = new Random();
        ID = ++idAnimals;
        this.NAME = NAMES[rd.nextInt(NAMES.length)]; //Random name
from the NAMES array
        this.age = age;
        this.attractiveness = attractiveness;
        this.PRICE = price;
        this.HETEROCHROMIA = (Objects.hash(NAME, age, attractiveness,
price, ID)% 635) < 1; //Animal chances to have heterochromia
        this.ALBINISM = (Objects.hash(NAME, age, attractiveness, ID)%
100) == 2; //Animal chances to have albinism
```

```
}

public double moneyWithAttractiveness() {
    return attractiveness * 50 + (1/(age+1)) * 20;
}

public int getId() {
    return ID;
}

public String getNAME() {
    return NAME;
}


public int getAge() {
   return age;
}

public void increaseAge() {
    age++;
}

public double getAttractiveness() {
    return attractiveness;
}

public void setAttractiveness(double attractiveness) {
    this.attractiveness = attractiveness;
}

public boolean getHeterochromia() {
    return HETEROCHROMIA;
}

public boolean getAlbinism() {
    return ALBINISM;
}

public static String[] getClasses() {
    return CLASSES;
}

public static String[] getCharateristics() {
    return CHARATERISTICS;
}

public static String[] getMutations(){
    return MUTATIONS;
}

public static String[] getNames() {
    return NAMES;
}

public int getPRICE() {
    return PRICE;
```

```
    }

    public int[] getTIMELIFE() {
        return TIMELIFE;
    }

    public void setTimeLife(int timeLifeLow, int timeLifeHigh) {
        TIMELIFE[0] = timeLifeLow;
        TIMELIFE[1] = timeLifeHigh;
    }

    public static String[] getCalendar() {
        return CHINESECALENDAR;
    }

    //Returns the chinese animal of that year
    public static String getChineseAnimal(int year) {
        return getCalendar()[year%12];
    }

    public double getMaintenance() {
        return maintenance;
    }

    public void setMaintenance(double value) {
        maintenance = value;
    }

    @Override
    public String toString() {
        return "Animal: " + getClass().getSimpleName() + " | Name: " +
NAME + " | Idade: " + age + " | Albino: " + getAlbinism() + " |
Heterocromia: " + getHeterochromia() + " | Preço: " + getPRICE() + " |
ID: " + getId();
    }
}
```

Classe Installations

```
package com.mycompany.zoo;

import java.util.ArrayList;

public class Installation {

    private static int idInstallations = 0;
    private final int ID;
    private final int CAPACITY; //Capacity of the installation
    private final ArrayList<Animal> ANIMALS; //Array with all the
animals in the installation
    private final int PRICE; //Price of the installation

    public Installation(int capacity, int price) {
        ID = ++idInstallations;
```

```
        this.CAPACITY = capacity;
        ANIMALS = new ArrayList<>();
        this.PRICE = price;
    }

    public int getCAPACITY() {
        return CAPACITY;
    }

    public int getID() {
        return ID;
    }

    public ArrayList<Animal> getANIMALS() {
        return ANIMALS;
    }

    public void addAnimals(Animal oneAnimal) {
        ANIMALS.add(oneAnimal);
    }

    public boolean verifyCapacity() {
        return ANIMALS.size() == CAPACITY;
    }

    public int getPRICE() {
        return PRICE;
    }

    @Override
    public String toString() {
        return "Installation " + ID + " | Capacity: " + CAPACITY + " |
Occupied: " + ANIMALS.size() + " | Price: " + PRICE;
    }
}
```

Classe Boi

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Boi extends Animal{

    public Boi(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(15, 20);
        setMaintenance(50);
    }


}
```

Classe Cabra

```java
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Cabra extends Animal{

    public Cabra(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(15, 18);
        setMaintenance(35);
    }


}
```

Classe Cao

```java
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Cao extends Animal{

    public Cao(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(10, 13);
        setMaintenance(40);
    }


}
```

Classe Cavalo

```java
package com.mycompany.zoo;

import static com.mycompany.zoo.Panthera.PRICE;

/**
*
* @author nunoo
*/
public class Cavalo extends Animal implements Equus {

    public Cavalo(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(25, 30);
        setMaintenance(55);
    }


    @Override
```

```
    public void abrirTerreno() {

    }

    @Override
    public int getPRICE() {
        return PRICE;
    }
}
```

Classe Chita

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Chita extends Animal {

    public Chita(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(10, 12);
        setMaintenance(48);
    }
}
```

Classe Cobra

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Cobra extends Animal{

    public Cobra(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(15, 30);
        setMaintenance(30);
    }
}
```

Classe Coelho

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Coelho extends Animal {
```

```
    public Coelho(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(8, 12);
        setMaintenance(15);
    }
}
```

Classe Dragão

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Dragao extends Animal{

    public Dragao(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(1000, 10000);
        setMaintenance(300);
    }
}
```

Classe Galo

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Galo extends Animal{

    public Galo(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(5, 10);
        setMaintenance(25);
    }
}
```

Classe jaguar

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Jaguar extends Animal implements Panthera {

    public Jaguar(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(12, 15);
```

```
    }
    @Override
    public void rugir() {
        System.out.print("O jaguar" + getNAME() + "rugiu.(Rawr!)");
    }

    @Override
    public int getPRICE() {
        return PRICE;
    }
}
```

Classe Leão

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Leao extends Animal implements Panthera {

    public Leao(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(8, 16);
        setMaintenance(60);
    }



    @Override
    public void rugir() {
       System.out.print("O leao" + getNAME() + "rugiu.(Rawr!)");
    }

    @Override
    public int getPRICE() {
        return PRICE;
    }
}
```

Classe Macaco

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Macaco extends Animal{

    public Macaco(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(20, 40);
        setMaintenance(42);
    }
```

```
}
```

Classe Porco

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Porco extends Animal{

    public Porco(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(15, 20);
        setMaintenance(45);
    }

}
```

Classe Rato

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Rato extends Animal{

    public Rato(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(2, 7);
        setMaintenance(20);
    }
}
```

Classe Tigre

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public class Tigre extends Animal implements Panthera {

    public Tigre(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(8, 10);
        setMaintenance(100);
    }

    /**
```

```
     *
     */
    @Override
    public void rugir() {
        System.out.print("O tigre" + getNAME() + "rugiu.(Rawr!)");
    }

    @Override
    public int getPRICE() {
        return PRICE;
    }

}
```

Classe Zebra

```
package com.mycompany.zoo;

import static com.mycompany.zoo.Panthera.PRICE;

/**
*
* @author nunoo
*/
public class Zebra extends Animal implements Equus {

    public Zebra(int age, double attractiveness, int price) {
        super(age, attractiveness, price);
        setTimeLife(20, 40);
        setMaintenance(30);
    }



    @Override
    public void abrirTerreno() {

    }

    @Override
    public int getPRICE() {
        return PRICE;
    }

}
```

Interface Equus

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public interface Equus {
```

```
    int PRICE = 400;

    public void abrirTerreno();

}
```

Interface Panthera

```
package com.mycompany.zoo;

/**
*
* @author nunoo
*/
public interface Panthera {

    int PRICE = 1000;

    public void rugir();}
```