Name: Utsav Budathoki

FAN: buda0027

StudentID: 2306084

TASK 1 - ANIMAL CLASS

1. What issues (if any) did you encounter with the program?

I did not encounter any issues with the program.

2. Describe any issues with the code as initially downloaded.

I did not encounter any issues with the code as initially downloaded.

3. What did you have to fix in order to compile and run the code successfully in IntelliJ?

I needed to update the program's configuration to be compatible with Java 17.

4. Explain how the first line of the provided toString() method works.

The first line uses a ternary operator `?` to append “Dead” in the `statusMessage` variable if the animal is dead. First it checks if the animal is dead or not by calling the `isDead()` method. If the `isDead()` method returns true, then “Dead” is appended in the `statusMessage`. Otherwise empty string is appended.

TASK 2 - ANIMAL SELECTION

1. How did you ensure that the user provided a positive integer for the number of animals?

There can be either valid or invalid input. The invalid input can be any text having alphabets, zero, and negative numbers. The valid input is a number greater than zero. So to keep track of input I have used the `isValidInput` boolean variable which is false at first. The loop breaks when the value of `isValidInput` is true.

There are three possible inputs. One of them is the invalid input like text. To handle this I have used try-catch if the code throws the `InputMismatchException`. The other one is the negative number or the zero. To handle this I have used an if statement to check if the number is greater than zero. The loop terminates when the user provides the positive number greater than zero.

2. How did you ensure that the user provided a unique name for each animal?

To achieve this I have used ArrayList data structure. Everytime a user enters a name, it is checked whether the name already exists or not by `animalNameList.contain(animalName).` `The name is taken only if it is unique.`

3. How did you ensure that the user provided a valid species for each animal?

I've used a boolean variable called `isValidSpecies` as a control variable for a loop, initially setting it to `false`. When the user enters a species name, I convert it to lowercase to ensure consistency. Inside a `switch` statement, I set `isValidSpecies` to `true` when the user enters one of the valid species names like "tiger," "panda," "monkey," "hippo," or "giraffe." If the user enters an invalid species name, the default case doesn't change the value of `isValidSpecies`, which prompts the user to reenter the correct species name.

TASK 3 - ITEM SELECTION

1. How did you check whether an item specified by the user for an animal was different from the most recent item that was given to that animal.

To achieve this, I have used an instance variable in the `Animal` class named `previousItem`. When the user enters the valid item, it is further checked that the item is the same as the previous one. If the item provided by the user is different from the previous one, the previous item is replaced by the new item and if the item is the same as the previous one the user is prompted with the message and ask to re-enter the different item.

2. The zoo has asked you to modify the program so that an item can only be given to an animal if it was not the same as either of the previous two items it was given. How would you need to change the code to achieve this?

To achieve this, we can replace the previousItem in Animal class with `ArrayList<String> lastTwoItem = new ArrayList<>()`. The function of lastTwoItem is to hold the value of the last two items given to the specific animal. We will add the item into the list when the list does not contain it. And when the size of the list reaches 2, we will remove the value at index at 0. By doing so, we will always have the latest two items in the arraylist.

```
if (Arrays.asList(validItems).contains(item)) {
   if (!animal.getLastTwoItems().contains(item)) {
       isValidItem = true;
       if (animal.getLastTwoItems().size() >= 2) {
           animal.getLastTwoItems().remove(0);
       }
       animal.getLastTwoItems().add(item);

       switch (item) {
           case "food" -> animal.giveFood();
           case "water" -> animal.giveWater();
           case "toy" -> animal.giveToy();
       }
   } else {
       System.out.println("You cannot give the same item as yesterday");
   }
} else {
   System.out.println("That is not a valid item");
}
```

TASK 4 - WEEK CYCLE

1. Describe what difference having the totalCost variable as static makes to your program.

Making totalCost static makes it easy to access the getter and setter with just the class name i.e. VirtualZoo. If totalCost is not static then we will need to create the object of the VirtualZoo class in every subclass of Animal (Tiger, Panda, Hippo, Giraffe, Monkey) to manipulate the getter and setter. Making totalCost static saved us from creating unnecessary objects in our program.

2. Describe what changes need to be made to your program in order for the user to manage multiple zoos with their own separate animals and costs.

The changes are listed down:

- We have to have a separate class "Zoo" where the attribute can be totalCost, name, listOfAnimal (ArrayList<Animal>).
- We also need to update the list of valid species to match the specific requirement of each zoo.
- Furthermore, we also need to create and implement classes for each animal type with their unique characteristics and behaviour (ie. implement the abstract method from Animal class).
- We need a method to ask the number of Zoo `askNumberOfZoo()` which returns the integer value.

TASK 5 - ANIMAL SPECIES

1. Is there an alternative way that we could determine the species of an Animal object without calling the getSpecies() method?

Yes, there is an alternative way that we could determine the species of an Animal object without calling the getSpecies() method. We can use the `instanceOf` operator.

```
if (animal instanceof Giraffe) {
    // operations
```

Reference: https://www.javatpoint.com/downcasting-with-instanceof-operator

2. What impact does making the `Animal` class an abstract class have on our program?

By making Animal class an abstract class, we can create an abstract method. By creating three abstract methods, we just need to declare it once in the Animal class and can have the different implementation on the extended class.

3. What difference would making the `Animal` class an interface, instead of an abstract class, have on our program?

By making Animal class as interface, we cannot use constructor, getter & setter, and instance variable in the interface since it only supports abstract methods and constant variables. All methods and variables need to be moved into all the subclasses which will just add more lines in the program.

4. The zoo has asked you to write a Graphical User Interface for the program you have written. Could it be used as is, or would you need to rewrite parts of the code to accomplish this? Discuss and explain.

Swing can be used to create the Graphical User Interface for the zoo. There can be a lot of other tools and frameworks for creating the GUI. We will need to rewrite part of the code because in the GUI, the user interaction is handled through graphical elements like button, text fields. However the business logic will remain the same.