CS 1150 Design Notebook Required Sections

Step 1: Problem Statement

Takens in values and types of objects into a polymorphic array. After all values are in the array display the array and each of the object's swim, run, and climb stats in a neat table. Then use the array to find what animal object has the best climb stat, display the animals found, and then also find what animal has the highest overall stats and display this animal's description.

Step 2: Understandings

- What I Know:
    - File reading
    - Arrays
    - Method
    - Classes and Objects
    - Casting
- What I Don't Know:
    - Interfaces, new to them,
    - Abstract classes, also new

Step 3: Pseudocode

```
Main:
    • Read the given file
        o First read number for array size
        o Go through the file and create Objects depending on species read
        o Put object into array using polymorphism
    • Call displayAnimals method only takes in one animal
        o Returns nothing
    • Call findClimbers method takes in the whole array
        o Returns an arrayList of Animals objects
    • Call findMostSkilled Method also takes in the whole array
        o Returns an integer

findMostSkilled:
    • Takes in polymorphic array and returns a number
    • For loop through the array to find what animal has the highest overall stats
      using the object's swim, run, and climb speeds
    • Return index of strongest animal
```

Step 4: Lesson Learned

I didn't realize I had to cast the object to its interface in order to use its methods and also at first I thought I could cast each animal to each interface but that doesn't work because not all of them have each interface requiring if statements.

Step 5: Code

```
//package cs1450;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

/*
Isaiah Hoffer
CS1450 (M/W)
2/13/25
Assignment 3
This assignment will take information from a file and create object to put into an array.
Use this array to display animals' stats and names and also find what animals are skilled in
Climbing and what animal is the best out of all of them. This assignment will use polymorphism,
arrays, arrayLists, interfaces, and abstract methods.
*/

public class HofferIsaiahAssignment3 {

	public static void main(String[] args) throws IOException {

		//Creating File Varialbe
		File fileName = new File("Animals.txt");

		//Allowing File To be Read
		Scanner fileRead = new Scanner(fileName);

		//Finding Length For Array
		int arrayLength = fileRead.nextInt();

		//Creating Polymorphic Array
		Animal[] animalObjArray = new Animal[arrayLength];

		//Reading File To Fill Array
		for(int i = 0; i < animalObjArray.length; i++ ) {

			//Creating Varialbe to Hold File Information
			String animalName = fileRead.next();
			String animalType = fileRead.next();

			int swimSpeed = fileRead.nextInt();
			int runSpeed = fileRead.nextInt();
			int climbSpeed = fileRead.nextInt();

			//Creating Ojbject Based On Species
			switch(animalType) {
				case "giraffe":

					animalObjArray[i] = new Giraffe(animalName, runSpeed);
					break;
				case "alligator":
```

```
                                        animalObjArray[i] = new Alligator(animalName, swimSpeed,
runSpeed);
                                        break;
                                case "sloth":

                                        animalObjArray[i] = new Sloth(animalName, swimSpeed,
climbSpeed);
                                        break;
                                case "monkey":

                                        animalObjArray[i] = new Monkey(animalName,runSpeed,
climbSpeed);
                                        break;
                                case "bear":

                                        animalObjArray[i] = new Bear(animalName, swimSpeed,
runSpeed, climbSpeed);
                                        break;
                        }//Switch

                }//For

                //Displaying Array
                System.out.printf("-----------------------------------------------------\n"
                                + "\t\tAll Animals In Array\t\t\n"
                                + "-----------------------------------------------------\n");
                for(int i = 0; i < animalObjArray.length; i++) {

                        displayAnimal(animalObjArray[i]);
                }//For

                //Finding Animals That Can Climb
                ArrayList<Animal> animalClimberList = findClimbers(animalObjArray);

                //Displaying Climbers
                        //Pretext
                System.out.printf("\n-----------------------------------------------------\n"
                                + "\t\tAnimals That Can Climb\t\t\n"
                                + "-----------------------------------------------------\n\n"
                                + "Name\t\tSpecies\t\tClimbing Speed\t\t\n"
                                + "-----------------------------------------------------\n");
                for(int i = 0; i < animalClimberList.size(); i++) {

                        Climber animalCasted = (Climber) animalClimberList.get(i);

                        System.out.printf("%s \t\t%s\t\t%d\n",
                                        animalClimberList.get(i).getName(),
animalClimberList.get(i).getSpecies(), animalCasted.getClimbSpeed());

                }//For
```

```
			//Finding Most Skilled Animal
			int indexOfStrongestAnimal = findMostSkilled(animalObjArray);

			//Displaying Strongest Animal
				//Pretext
			System.out.printf("----------------------------------------------------\n"
						+ "\t\tMost Skilled Animal\t\t\n"
						+ "----------------------------------------------------\n");

			//Animal Name
			System.out.printf("%s the %s says
%s!!!\n",animalObjArray[indexOfStrongestAnimal].getName(),
animalObjArray[indexOfStrongestAnimal].getSpecies(),
						animalObjArray[indexOfStrongestAnimal].makeNoise());

			//Getting Animal's Values, Also Makes Sure Animals has those Values
			if(animalObjArray[indexOfStrongestAnimal] instanceof Swimmer) {

				Swimmer animalSwimmer = (Swimmer)
(animalObjArray[indexOfStrongestAnimal]); //Casting Object To Get Interfaces For Animal Stats

				System.out.printf("Swimming Speed: %d\n",
animalSwimmer.getSwimSpeed());
			}//If

			if(animalObjArray[indexOfStrongestAnimal] instanceof Runner) {

				Runner animalRunner = (Runner) (animalObjArray[indexOfStrongestAnimal]);
//Casting Object To Get Interfaces For Animal Stats

				System.out.printf("Running Speed: %d\n", animalRunner.getRunSpeed());
			}//If

			if(animalObjArray[indexOfStrongestAnimal] instanceof Climber) {

				Climber animalClimber = (Climber)
(animalObjArray[indexOfStrongestAnimal]); //Casting Object To Get Interfaces For Animal Stats
				System.out.printf("Climbing Speed: %d\n", animalClimber.getClimbSpeed());
			}//If


		}//main

//Display Animals Method
//Takes In on Object From Array And Disaplys Animal's Info
public static void displayAnimal(Animal animal) {

		System.out.printf("\n%s the %s says %s!!!\n",animal.getName(), animal.getSpecies(),
animal.makeNoise());
```

```
		if(animal instanceof Swimmer) {

			Swimmer animalSwimmer = (Swimmer) (animal); //Casting Object To Get Interfaces
For Animal Stats

			System.out.printf("Swimming Speed: %d\n", animalSwimmer.getSwimSpeed());
		}//If

		if(animal instanceof Runner) {

			Runner animalRunner = (Runner) (animal); //Casting Object To Get Interfaces For
Animal Stats

			System.out.printf("Running Speed: %d\n", animalRunner.getRunSpeed());
		}//If

		if(animal instanceof Climber) {

			Climber animalClimber = (Climber) (animal); //Casting Object To Get Interfaces For
Animal Stats
			System.out.printf("Climbing Speed: %d\n", animalClimber.getClimbSpeed());
		}//If


	}//displayAnimal Method

	//Method to Use Animal Array to Find ALL Climbers
	public static ArrayList<Animal> findClimbers(Animal[] animals) {

		//Creating ArrayList
		ArrayList<Animal> animalClimbersList = new ArrayList<>();

		for(int i = 0; i < animals.length; i++) {

			if(animals[i] instanceof Climber) {

				animalClimbersList.add(animals[i]);

			}//If


		}//For

		return animalClimbersList;

	}//findClimbers Method

	//Method To Find Most Skilled Animal
	public static int findMostSkilled(Animal[] animals) {

		int bestAnimalIndex = 0;
```

```
        int highestStats = 0;

        //Finding Best Animal
        for(int i = 0; i < animals.length; i++) {

            int animalStats = 0;

            //Checking What Stats To Add
            if(animals[i] instanceof Swimmer) {

                Swimmer animalSwimmer = (Swimmer) (animals[i]);

                animalStats += animalSwimmer.getSwimSpeed();
            }//If

            if(animals[i] instanceof Runner) {

                Runner animalSwimmer = (Runner) (animals[i]);

                animalStats += animalSwimmer.getRunSpeed();
            }//If

            if(animals[i] instanceof Climber) {

                Climber animalSwimmer = (Climber) (animals[i]);

                animalStats += animalSwimmer.getClimbSpeed();
            }//If

            //Looking If Value Is Bigget
            if(animalStats > highestStats) {

                bestAnimalIndex = i;
                highestStats = animalStats;

            }//If


        }//For


        return bestAnimalIndex;

    }//findMostSkilled Method

    }//Class

    /*******************************
                  CLASSES
    *******************************/
```

```
//Abstract Animal Class
//Sets and Gets animals Name and Species
	//SubClasses: Alligator, Bear, Giraffe, Moneky, Sloth
abstract class Animal {

	//Initalizing private Data
	private String name;
	private String species;

	//Method to set name
	public void setName(String name) {

		this.name = name;
	}//setName

	//Method to set Species
	public void setSpecies(String species) {

		this.species = species;
	}//setSpecies

	//Method to get name
	public String getName() {

		return name;
	}//getName

	//Method to get Species
	public String getSpecies() {

		return species;
	}//setSpecies


	//Retruns noise of Animal
	public abstract String makeNoise();


}//Animal Abs. Class

//Alligator Class
class Alligator extends Animal implements Swimmer, Runner {

	//Priavte Data Fields
	private int swimSpeed;
	private int runSpeed;

	//Alligator Constructor
	public Alligator(String name, int swimSpeed, int runSpeed) {

		//Setting Values
		this.swimSpeed = swimSpeed;
		this.runSpeed = runSpeed;

		super.setName(name);
		super.setSpecies("Alligator");

	}//Alligator Cons.

	public String makeNoise() {

		return "Crunch";

	}//makeNoise

	@Override
	//Getter for SwimSpeed
	public int getSwimSpeed() {

		return swimSpeed;
	}//getSwimSpeed Method

	@Override
	//Getter for RunSpeed
	public int getRunSpeed() {

		return runSpeed;
	}//getRunSpeed Value


}//Alligator Class

//Bear Class
class Bear extends Animal implements Swimmer, Runner, Climber {

	//Priavte Data Fields
	private int swimSpeed;
	private int runSpeed;
	private int climbSpeed;

	//Bear Constructor
	public Bear(String name, int swimSpeed, int runSpeed, int climbSpeed) {

		//Setting Values
		this.swimSpeed = swimSpeed;
		this.runSpeed = runSpeed;
		this.climbSpeed = climbSpeed;

		super.setName(name);
		super.setSpecies("Bear");

	}//Bear Cons.
	public String makeNoise() {

		return "Growl";

	}//makeNoise

	@Override
	//Getter for SwimSpeed
	public int getSwimSpeed() {

		return swimSpeed;
	}//getSwimSpeed Method

	@Override
	//Getter for RunSpeed
	public int getRunSpeed() {

		return runSpeed;
	}//getRunSpeed Value

	@Override
	//Getter for ClimbSpeed
	public int getClimbSpeed() {

		return climbSpeed;
	}//getClimbSpeed Method



}//Bear Class

//Giraffe Class
class Giraffe extends Animal implements Runner {

	//Priavte Data Fields
	private int runSpeed;

	//Bear Constructor
	public Giraffe(String name, int runSpeed) {

		//Setting Values
		this.runSpeed = runSpeed;

		super.setName(name);
		super.setSpecies("Giraffe");

	}//Giraffe Cons.

	public String makeNoise() {
		return "Bleat";

	}//makeNoise

	@Override
	//Getter for RunSpeed
	public int getRunSpeed() {

		return runSpeed;
	}//getRunSpeed Value


}//Giraffe Class

//Moneky Class
class Monkey extends Animal implements Runner, Climber {

	//Priavte Data Fields
	private int runSpeed;
	private int climbSpeed;

	//Bear Constructor
	public Monkey(String name, int runSpeed, int climbSpeed) {

		//Setting Values
		this.runSpeed = runSpeed;
		this.climbSpeed = climbSpeed;

		super.setName(name);
		super.setSpecies("Moneky");

		}//Bear Cons.

	public String makeNoise() {

		return "Screech";

	}//makeNoise

	@Override
	//Getter for RunSpeed
	public int getRunSpeed() {

		return runSpeed;
	}//getRunSpeed Value

	@Override
	//Getter for ClimbSpeed
	public int getClimbSpeed() {
		return climbSpeed;
	}//getClimbSpeed Method


}//Moneky Class

//Sloth Class
class Sloth extends Animal implements Swimmer, Climber {

	//Priavte Data Fields
	private int swimSpeed;
	private int climbSpeed;

	//Sloth Constructor
	public Sloth(String name, int swimSpeed, int climbSpeed) {

		//Setting Values
		this.swimSpeed = swimSpeed;
		this.climbSpeed = climbSpeed;

		super.setName(name);
		super.setSpecies("Sloth");

	}//Sloth Cons.

	public String makeNoise() {

		return "Squeak";

	}//makeNoise

	@Override
	//Getter for SwimSpeed
	public int getSwimSpeed() {

		return swimSpeed;
	}//getSwimSpeed Method

	@Override
	//Getter for ClimbSpeed
	public int getClimbSpeed() {

		return climbSpeed;
	}//getClimbSpeed Method


}//Sloth Class



/*******************************
```

```
                        INTERFACES
*********************************/

//Interface to return Animal's Swim Value
interface Swimmer {

            abstract public int getSwimSpeed();


}//Swimmer Interface


//Interface to return Animal's Climb Value
interface Climber {

            abstract public int getClimbSpeed();


}//Climber Interface

//Interface to return Animal's Run Value
interface Runner {

            abstract public int getRunSpeed();


}//Runner Interface
```