**CS1150 HW09 Polymorphism and Files**

Due Date: **See Canvas**

Purpose: Demonstrate the use inheritance and polymorphism and read from a file

Effort: **Individual (**[**Academic Integrity**](https://docs.google.com/document/d/e/2PACX-1vTwfw8GoGwerw-O8dkH-mkjhuLdpb_q3FIaMrY2YiGgdF2h_u2VLkfsAHRIHB31TEWu5lpfTzO7G3rG/pub)**)**

Points: 80

Deliverables: Upload **lastNameFirstNameHW010.java** file and learning/reflection document.

**Assignment Description**

This assignment is an opportunity to work with objects using inheritance and polymorphism. Write a program that creates one superclass (parent), several subclasses (child), and use polymorphism. Write code to create the following class hierarchy in your program.

Animal



Bear Elephant Monkey Sloth

| **Superclass**   * Animal | **Subclasses of Animal**   * Bear * Elephant * Monkey * Sloth |
| --- | --- |

The code will perform actions on these objects and display information proving you have properly used inheritance and polymorphism. Use the provided file [**Animals.txt**](https://drive.google.com/file/d/1K8QJ7w0KJsq2qX1d96ibiJkKlwwLznTo/view?usp=share_link) to test your code.

**Specifications**

Read these specifications and output to get an overview of the assignment. When writing code, **first** create the classes.

* Before writing the code in main, build as much of each class as possible
* You need to have at least the constructor of each class created before you can write the code to create animals in main.

1. Create a Java class **LastNameFirstNameHW08** within that project calledwithin the **CS1150HW** project. Place all classes in one Java file.
2. Follow [Problem Solving Using Software Development Life Cycle](https://docs.google.com/presentation/d/e/2PACX-1vSC4tM0BsOaYjRmd_emNeHtKrWaExKXfvvLxQC10rCt77CPT_WvF_s3W--o_yuj8XczYGo7qHxgjMIq/pub?start=false&loop=false&delayms=3000&slide=id.ge6997ea817_0_125)
3. Write the superclass **Animal** (see Classes section below for details)
4. Write 4 Animal subclasses: **Bear, Elephant, Monkey, Sloth** (see Classes section below for details)
5. Write a **test program** (i.e. main) that:
   1. Open the test file **Animals.txt** for reading.
   2. Creates a **polymorphic array** to store the animals.
      1. The 1st value in the file indicates how big to make the array.
      2. Be sure to use a regular array, NOT an array list.
   3. Uses a loop to fill the array with each animal’s information read from a file
      1. Instead of asking the user for information, you will read information from a file.
      2. After reading the 1st value (array size), read the remaining values to create the animal objects and place them into the array

For each line in the file:

* + - * 1. Read the type (bear, elephant,etc.), name, food, weight, sleep, and location of the animal.
        2. Create the **specific** animal object

The ***type*** string indicates what object to create.

Note that ***type*** is not sent to the constructor.

* + - * 1. Place the animal object into the polymorphic **Animal** array.
    1. Be sure to read the strings using ***next()*** and weight and sleep values using ***nextInt()***

* 1. **AFTER** all lines in the file have been read and all animal objects are in the array:
     1. Create a 2nd loop to iterate through the array and:
        1. Display ***type*** (Bear, Elephant, etc.) of animal

Animal[0] is a Sloth

* + - * 1. Note that ***type*** is not an instance variable in Animal
        2. Instead, we can use the ***instanceof*** operator to determine which **object** is in the current location in the array and display it’s type
        3. For more details. read Liang, **section 11.9 Casting Objects and the instanceof Operator**

Listing 11.7 shows the use of **instanceof** – ignore the information about casting.

Use **instanceof** to determine if the current object is a Bear, Elephant, Monkey, or Sloth.

Display an indication of what instance the current object in the array is:

* + - 1. Display name, weight, sleep and location of animal
         1. Call ***toString*** to display these values
      2. Call the following methods:

eat

sleep

swim

* + 1. See output section at end of document for example output

1. Test file (**Animals.txt**) information:
   1. Run your code on the provided file **Animals.txt** to test your code.
   2. This is a test file, so **DO NOT** assume that your code should work for only 7 animals in the order specified in the file. The file we use to test will be updated.
   3. The file contains the following animals:

| **Animal** | **Name** | **Food** | **Weight** | **Sleep** | **Location** |
| --- | --- | --- | --- | --- | --- |
| Sloth | Ace | Leaves | 12 | 15 | Urban Jungle |
| Bear | Po | Bamboo | 550 | 20 | Asian Passage |
| Monkey | Rafiki | Fruit | 27 | 17 | Lost Forest |
| Sloth | Sid | Leaves | 15 | 12 | Urban Jungle |
| Elephant | Titan | Grass | 12500 | 4 | Elephant Odyssey |
| Monkey | Louie | Fruit | 14 | 16 | Lost Forest |
| Bear | Baloo | Honey | 1050 | 18 | Asian Passage |

* 1. The 1st line in the file is an integer value representing the number of animals in the file
  2. The remaining lines contain details for each animal
     1. There will be **one animal** for each line in the file.
     2. The format of each **animal line** is as follows:

Type (tells you what object to create – it is **NOT stored** as an instance variable in object)

**Sloth Ace Leaves 12 15 Urban Jungle**

Name Food Weight Sleep Location (**store** these 5 values in instance variables)

* 1. The **Animal.txt** file that is provided looks like this:

7 Number of animals

Sloth Ace Leaves 12 15 Urban Jungle

Bear Po Bamboo 550 20 Asian Passage

Monkey Rafiki Fruit 27 17 Lost Forest Each line provides the

Sloth Sid Leaves 15 12 Urban Jungle details for one animal

Elephant Titan Grass 12500 4 Elephant Odyssey

Monkey Louie Fruit 14 16 Lost Forest

Bear Baloo Honey 1050 18 Asian Passage

**Classes**

**Animal Class**

* Description
  + Class that represents an animal.
  + Superclass in the hierarchy.
* Data Fields
  + **name** – String
  + **food** - String
  + **weight** – int
  + **sleep** - int
  + **location** - String
* Public Methods
  + Constructor: ***public Animal(String name, String food, int weight,***

***int sleep, String location)***

* + - Creates an Animal by initializing all private data fields to incoming values
  + Getter for each instance variable - **name**, **food**, **weight**, **sleep**, and **location**
  + Setters – none (this is being done on purpose so do **NOT** include setters)
  + The following additional public methods. The only code in these methods is a statement that displays the action the method represents for Animal (e.g. "Animal is eating")
    - ***public void eat()***
    - ***public void sleep()***
    - ***public void swim()***

**Bear, Elephant, Monkey, Sloth** **Subclasses**

* Description
  + Each class represents a specific animal.
  + Each of these classes must be a subclass of Animal
* Data Fields
  + None
* Public Methods
  + Constructor – each subclass must have a constructor that:
    - Sets instance variables to incoming values by calling constructor in parent.
    - Remember: **super** calls the constructor in parent class.
    - Note that **type** is not sent to the constructor.
    - For example, the Bear constructor looks like:
      * ***public Bear(String name, String food, int weight,***

***int sleep, String location)***

* + - Do the same for the constructor in the Elephant, Monkey, and Sloth subclasses.
  + Getters/Setters - None
  + Each subclass **MUST** override certain methods in the Animal class.

Note: what is overridden is different for each subclass.

* + 1. The table indicates which methods inherited from Animal must be overridden.
       1. "Yes" in column - **MUST** override method in specific subclass.
       2. "No" in column - **DO NOT** override method in specific subclass.
       3. For example, for **Monkey**, must override the eat and swim methods:

@Override

**public** **void** eat() { // Add code here }

@Override

**public** **void** swim() { // Add code here }

| **Animal** | **eat()** | **sleep()** | **swim()** |
| --- | --- | --- | --- |
| Bear | **Yes** | **Yes** | **Yes** |
| Elephant | **No** | **Yes** | **No** |
| Monkey | **Yes** | **No** | **Yes** |
| Sloth | **No** | **No** | **No** |

* + The only code in the overridden methods in each subclass is:
    - A statement to display the action for the method (e.g. "Bear is swimming")
    - For ***eat*** and ***sleep*** methods include the food and sleep hours for the animal
  + Finally, each subclass **MUST** override the ***toString()*** method to display the object's:
    1. Name
    2. Weight
    3. Sleep
    4. Location

**Sloth: Name: Ace - Weighs: 12.0 lbs - Sleeps: 15.0 hours - Location: Urban Jungle**

* + For all overridden methods, you must include the @Override

**Output**

To prove your program is using inheritance and polymorphism correctly, your output **MUST** look like:

**Output - Example**

Animal[0] is a Sloth

Sloth: Name: Ace - Weighs: 12 lbs - Sleeps: 15 hours - Location: Urban Jungle

Animal is eating

Animal is sleeping - do not disturb

Animal is swimming

Animal[1] is a Bear

Bear: Name: Po - Weighs: 550 lbs - Sleeps: 20 hours - Location: Asian Passage

Bear is eating Bamboo

Bear is sleeping 20 hours

Bear is swimming

Animal[2] is a Monkey

Monkey Name: Rafiki - Weighs: 27 lbs - Sleeps: 17 hours - Location: Lost Forest

Monkey is eating Fruit

Animal is sleeping - do not disturb

Monkey is swimming

Animal[3] is a Sloth

Sloth: Name: Sid - Weighs: 15 lbs - Sleeps: 12 hours - Location: Urban Jungle

Animal is eating

Animal is sleeping - do not disturb

Animal is swimming

Animal[4] is a Elephant

Elephant: Name: Titan - Weighs: 12500 lbs - Sleeps: 4 hours - Location: Elephant Odyssey

Animal is eating

Elephant is sleeping 4 hours

Animal is swimming

Animal[5] is a Monkey

Monkey Name: Louie - Weighs: 14 lbs - Sleeps: 16 hours - Location: Lost Forest

Monkey is eating Fruit

Animal is sleeping - do not disturb

Monkey is swimming

Animal[6] is a Bear

Bear: Name: Baloo - Weighs: 1050 lbs - Sleeps: 18 hours - Location: Asian Passage

Bear is eating Honey

Bear is sleeping 18 hours

Bear is swimming

**Note**

* When a method on Animal is called the generic "Animal" text is displayed
* When an overridden method is called the name of specific animal (e.g. "Monkey") is displayed 



Animal[5] is a Monkey

Monkey Name: Louie - Weighs: 14 lbs - Sleeps: 16 hours - Location: Lost Forest

Monkey is eating Fruit

Animal is sleeping - do not disturb

Monkey is swimming

|  | This image of course will not be  in your output! I included because  sloths are funny/cute looking! |
| --- | --- |

## Learning and Reflections

Explain the following Object Oriented concepts in your own words. Include code snippets from your homework code in your explanation.

1. Inheritance

Inhereitance is a concept where a new class can be created which inherits the properties and methods of theparent class. The new class is called the Subclas. The subclass can modify the properties and methods of the parent, or superclass.

**class** Bear **extends** Animal {

@Override

**public** **void** eat() {

System.***out***.println(**this**.getClass().getSimpleName()+ " is eating " + **this**.getFood());

}

1. Polymorphism

Polymorphism is the way objects can have multiple forms or behaviors, it allows objects of different clases to be used interchangeably when they share a parent class.

System.***out***.println(animals[i].getClass().getSimpleName() + ": Name: " + GetName +

" - Weighs: " + GetWeight + " lbs - Sleeps: " + GetSleep + " hours - Location: " + GetLocation);

1. Dynamic Binding

Dynamic binding is determining th proper method or function to call during runtime, ased on the type of object being used.

animals [i]