Lab 8: Inheritance

Bugs

Purpose

- Practice working with multiple classes
- Practice creating objects and calling methods
- Practice inheritance, polymorphism, and casting of objects

Description of Task

This lab comprises of two portions, a programming and a question portion. Complete the implementation program, then use it to answer the questions. **Please read the deliverables section and the rubric!**

Programming

Create the following classes according the descriptions provided:

Bug: The Superclass

Bug has the following private attributes:

- String species, which is the name of the bug species.
- String color, which is the color of the bug.

Bug has the following constructors:

- Bug
  - Parameters: None
  - This default constructor calls the parameterized constructor, with the species being “bug”, and the color being “tan”.
- Bug(String color, String species)
  - Parameters: String color, String species
  - This constructor sets the color and species attributes to the parameters that are passed to it.

Bug has the following public methods:

- String noise()
  - Parmeters: none
  - Returns a String that is the default sound of a bug, which is “blop”
- String toString()
  - Overrides toString() from Object
  - Parameters: none
  - Returns a String that concatenates the species, color, and noise (obtained by calling the method noise()), and returns it. So, for the default Bug, it should say “a tan bug that goes blop.”
- Setters and getters for attributes species (4 methods total).

Class Arachnid, extends Bug

Arachnid has the following private attributes:

- (Arachnid has no private attributes)

Arachnid has the following constructors:

- Arachnid()
  - Parameters: none

    - This default constructor calls the other parameterized constructor using “this”, with the color being “brown” and the species being “Arachnid”.
- Arachnid(String color, String species)
    - Parameters: two Strings, one for color and one for species
    - This constructor calls the super constructor with the appropriate parameters.

Arachnid has the following public methods:

- String toString(), overrides Bug’s toString.
    - Parameters: none
    - Return value: String
    - This method should return a string representing an Arachnid. It should clearly state this object is an Arachnid. It should use super’s toString() method, but prepend “Arachnid is ” to the string.
- String noise(), overrides Bug’s noise()
    - Parameters: None
    - Returns the noise an Arachnid makes, which is “scuttle”;
- int numberOfLegs()
    - Parameters: None
    - Returns the number of legs an Arachnid has, which is 8.

Class Insect, extends Bug

Insect has the following private attributes:

- boolean canFly

Insect has the following constructors:

- Insect()
    - Parameters: none
    - This default constructor calls the other parameterized constructor using “this”, with the color being “red” and the species being “Insect”, and the canFly being false.
- Insect(String color, String species, boolean canFly)
    - Parameters: two Strings, one for color and one for species, and a boolean
    - This constructor calls the super constructor with the appropriate parameters.
    - It also sets the canFly attribute to the parameter given to it.

Insect has the following public methods:

- String toString(), overrides Bug’s toString.
    - Parameters: none
    - Return value: String
    - This method should return a String representing an Insect. It should clearly state this object is an Insect. It should use super’s toString() method, but prepend “Insect is ” to it, and append if it can or cannot fly after it.
- String noise(String additionalNoise), overloads Bug’s noise()
    - Parameters: String additionalNoise, which is another noise to add to the default noise.
    - Returns a String of the default noise concatenated to the additional noise.
    - This method should call noise(), overriden from Bug, and concatenate the parameter additionalNoise to its return value. Then, return the combined string.
- String noise(), overrides Bug’s noise
    - Parameters: None
    - Returns “buzz”, the default sound of a Insect.
- int numberOfLegs()
    - Parameters: None
    - Returns the number of legs an Insect has, which is 6.

Class Bee, extends Insect

Bee has the following private attributes:

- (Bee has no private attributes)

Bee has the following constructors:

- Bee()
  - Parameters: none
  - This default constructor calls the other parameterized constructor using “this”, with the color being “yellow” and the species being “Bee”.
- Bee(String color, String species)
  - Parameters: two Strings, one for color and one for species
  - This constructor calls the super constructor with the appropriate parameters. It should hardcode canFly to true when calling the super constructor.

Bee has the following public methods:

- String toString(), overrides Insects’s toString.
  - Parameters: none
  - Return value: String
  - This method should return a string representing a Bee. It should clearly state this object is a “Bee”. It should use the getters from the super class and/or itself to build the String containing all attributes. It should also call noise(String additionalNoise) with the parameter “whizz” and include that in the string. Note that because the superclass toString() method using noise() without parameters, you cannot just call the super class toString() and add onto it. You’ll have to build up a string similar to in Bug’s toString(). Make sure you also include whether it can fly or not.
- void harvestHoney()
  - Parameters: none
  - Returns: nothing
  - This method just prints out “Harvesting honey!”.

Class Ant, extends Insect

Ant has the following private attributes:

- (Ant has no private attributes)

Ant has the following constructors:

- Ant()
  - Parameters: none
  - This default constructor calls the other parameterized constructor using “this”, with the color being “red” and the species being “Ant”.
- Ant(String color, String species)
  - Parameters: two Strings, one for color and one for species
  - This constructor calls the super constructor with the appropriate parameters. It should hardcode canFly to false when calling the super constructor.

Ant has the following public methods:

- String toString(), overrides Insects’s toString.
  - Parameters: none
  - Return value: String
  - This method should return a string representing a Ant. It should clearly state this object is a “Ant”. It should use the getters from the super class and/or itself to build the String containing all attributes. It should also call noise(String additionalNoise) with the parameter “dink” and include that in the string. This one will be very similar to Bee.

- void buildAntHill()
  - Parameters: none
  - Returns: nothing
  - This method just prints out “Building ant hill!”.
- int numberOfLegs(int injured), overloads Insects numberOfLegs()
  - Parameters: a int denoting if the ant is injured and how many legs it has lost
  - Returns: the value numberOfLegs obtained from the super class numberOfLegs() method, minus the injured int.

Class Main, the main class

Main has the following private attributes:

- (Main has no private attributes)

Main has the following public constructors:

- (Main has no constructors)

Main has the following public static methods:

- main(String[] args)
  - The main method creates bugs, stores them into a list, and then prints them out.
    1. Create the following bugs:
       1. A default Bug.
       2. A Bug that is a purple millipede.
       3. A default Arachnid.
       4. An Arachnid that is a brown tarantula.
       5. A default Insect.
       6. An Insect that is a blue beetle that can’t fly.
       7. A default Bee.
       8. A Bee that is a yellow hornet.
       9. A default Ant.
       10. An Ant that is a black leafcutter.
    2. Create an ArrayList that can hold Bugs. Loop through the list and print out each Bug (notice dynamic binding at work here). If the bug is a Bee, call harvestHoney() on it. If it is an ant, call buildAntHill() on it. Keep a sum of the number of legs of all the bugs, by calling numberOfLegs() on each bug. Note that Bug does not have numberOfLegs, so you will have to make sure it is at least an Insect or Arachnid (and downcast appropriately). If the bug is an Ant, call numberOfLegs(1), using the overloaded numberOfLegs the Ant class has.

Example Program Output

```
a tan bug that goes blop
a purple millipede that goes blop
Arachnid is a brown arachnid that goes scuttle
Arachnid is a brown tarantula that goes scuttle
Insect is a red insect that goes buzz and it cannot fly.
Insect is a blue beetle that goes buzz and it cannot fly.
Bee is a yellow Bee that goes buzzwhizz and it can fly.
Harvesting honey!
Bee is a yellow hornet that goes buzzwhizz and it can fly.
Harvesting honey!
Ant is a red ant that goes buzzdink and it cannot fly.
Building ant hill!
Ant is a black leafcutter that goes buzzdink and it cannot fly.
Building ant hill!
Total number of legs: 50
```

Questions

1. If we have the following code:

```
Bug bug = new Bug();
Bug spider = new Arachnid("red", "spider");
Bug grasshopper = new Insect();
Bee wasp = new Bee("yellow", "wasp");

System.out.println(wasp instanceof Bug);
System.out.println(wasp instanceof Insect);
System.out.println(spider instanceof Insect);
System.out.println(grasshopper instanceof Arachnid);
```

What will be printed? Explain your answer.

a. All true
b. All false
c. False, true, false, true
d. True, false, true, true
e. True, true, false, false
f. None of the above

2. If we have the following code:

```
Bee wasp = new Bee();
System.out.println(wasp.numberOfLegs());
System.out.println(wasp.getColor());
```

What will the output be? Explain your answer.

a. 6, tan
b. 6, yellow
c. Compile time error
d. Runtime error

3. If we have the following code, what is going to happen? Explain your answer.

```
Bug bug = new Bug();
Bug spider = new Arachnid();
Bug bee = new Bee();
Ant ant = new Ant("red", "fire");
Bug[] ar = { bug, spider, bee, ant };
for(Bug it: ar){
    if( it instanceof Insect){
        System.out.println(((Insect)it).numberOfLegs());
    }
}
```

a. 6, 8, 6, 6
b 6, 6
c. Nothing will be printed.

d. Compiler error
e. Runtime error

4. If we have the following code, what will happen? Explain your answer.

```
Monster zero = new Ghost("Zero");
Haunt h = (Haunt)zero;
Haunt tom = new Ghost("Tom Joad");
```

a. No errors will occur.
b. Runtime error on line 3, compile error on line 2.
c. Compile error on line 1.
d Compile error on Line 3.
e. Compile error on line 3, runtime error on line 2.

5. If we have the following code:

```
Bug ant = new Ant();
System.out.println((Ant)ant.numberOfLegs());
```

This code will error. What is the error, why does it happen, and how can we fix this code to output 6?

Helpful Tips

Getting Help
Please email me if you need help. Please do not wait until the last minute; get started early so there's plenty of time to help out.

Follow the criteria outlined in the slides when asking questions. Your questions should include:

- What kind of bug is it? Compile-time, run-time, or logic?
- What was the last thing you did before this error occurred? What were the last lines of code that you wrote, or what is the test case that fails?
- If the bug occurs at compile time or run time, what is the error message, and what part of the program does it indicate?
- What have you tried, and what have you learned?

Compiling Multiple Java Files at Once
Since we now have multiple files for our classes classes, you can compile them all at once with the command:

```
javac *.java
```

You can also compile them one at a time if you wish. Run the program with:

```
java Main
```

Use Resources for Guidance
Everything in this lab is demonstrated at depth in the textbook. Use resources as you code; no one is expected to memorize any of this stuff. Finding good resources and learning to use them as a guide is a fundamental skill of programming.

Deliverables

Submit the following artifacts to Canvas (separately, please, not zipped):

- All .java files for your program. Remember, the filename must match the class name.
- An answer document that contains the answers to the question section. Please number your answers correctly.
- A .rtf or .txt file with a reflection. The reflection should contain a few paragraphs covering the following:
  - What went well in this lab? What was easy and made sense?
  - What didn't go well? What was difficult or your struggled with?
  - What did you learn from this lab? What was the most valuable piece?
- **Note**: No .class files required for this lab submission.

Rubric (100 Points)

Programming Section (70 Points)

There are 5 bug classes. Each class will be graded on the following, 12 points a piece:
1 points (Class attributes)
The class has the correct attributes at the correct visibility levels.

3 points (Class constructors)
The constructors have the proper parameters and initialize a class instance correctly. They call super constructors and their own constructors as described in the document.

5 (Class other methods, overrides, and overloads, setters/getters)
Other class methods, like noise(), numberOfLegs(), harvestHoney(), and buildAntHill() are implemented correctly, overriding and overloading where described. The Bug class has setters and getters for its private attributes

3 (toString methods)
The toString methods are @Overrides, and they output all of the information specified in the problem description. They call the super class version of toString when appropriate.

10 Points (Main Method)
The main method constructs a list of Bugs using an array list, and outputs the correct information about the bugs. It uses instanceof and downcasting to convert the generic super class (Bug) into the specific class of the object. It counts the proper number of legs of all Insects and Arachnids and their subtypes.

Questions Section (20 Points)

4 Points (Each Question)
Each question will be 8 points each, provide explanations or show your work for partial credit, where applicable.

Reflection (10 Points)

5 Points (Completeness)
The reflection covers the required topics; what went well, what didn't go well, what was learned.

5 Points (Quality)
The reflection was written in complete sentences. It features multiple sentences, is coherent, and thoughtful.