COSC1337 Programming Fundamentals II/ ITSE2357 Advanced Object-Oriented Programming

Grade: Great job! 8/10. Please see comments below.

If one of the following is true, you will NOT get credits.

☐The project is late.
☐ The algorithm or class design is missing.
☐The project has syntax errors.
☐ There are no comments (Javadoc format required) at all in your code.
☐Wrong files are submitted.
☐ A project is a copy and modification of another student's project. (Both will receive 0.)

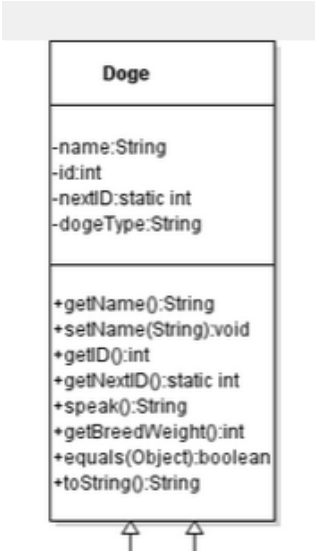
Software Development Project Rubric: Design is worth 20%; the rest is worth 80%.
Analysis
Note: There will be no credit if the software doesn't meet customer specification at all.
Does the software meet the exact customer specification?
Does the software read the exact input data and display the exact output data as they are shown in sample runs?
Does each class include all corresponding data and functionalities?

Design (-1)
Note: There will be no credit if the design is missing.
Is the design (a UML class diagram) an efficient solution?
Is the design created correctly?

When a complete design is given by a customer / designer, don't change it unless it is agreed by the customer/ designer. In this case, doge type is not needed since each dog type is already is defined as a sub type.

The name of each type must not be changed either since it is part of the requirements as well.

The basic rule of software development is to follow instructions to make what customers want.



Code
Note: There will be no credit if there are syntactic errors.
Are there errors in the software?
Are code conventions and name conventions followed?
Does the software use the minimum computer resource (computer memory and processing time)?
Is the software reusable?
Doge:
Remove dog type along with its setter and getter.

By conventions, a method name should start with a lowercase letter.
public abstract int ~~BreedWeight()~~ getBreedWeight();

Debug
Are there bugs in the software?
Documentation (-1)
Note: There will be no credit if comments are not included.
Are there enough comments included in the software?
Class comments must be included before a class header.
Method comments must be included before a method header.
More comments must be included inside each method.
All comments must be written in Javadoc format.

Comments are equally important.

When your classes are reused, only comments are shown for each class. By reading the class comments, people have no idea about the purpose of this class. How do they use it if they don't know what a class does?

```
/**
 * @author Julian
 * @version 1.0
 */
package doge;

public abstract class Doge implements Comparable<Doge>
```

{

Instance variables/static variables describe characteristics. Comments for variables are a phrase, and don't start with a verb since this variable does not create any characteristic by representing it.

```
//Creates a string for Name // The name of this dog
private String name;

// Creates an int for ID
private int id;

//Generates a nextID integer
private static int nextID = 100;

//Creates a type for the doge.
private String dogeType;
```

When document a method, start with a verb in the third tone to describe the purpose of this method. In addition to it, every formal parameter if any and the method return value if any must be documented using proper tags.
The comments for a formal parameter contain three parts:
@param TheNameOfParameter DescriptionOfParameter
The comments for a return value contain two parts:
@return DescriptionOfReturnValue

```
/**
 *  Constructs this dog with an empty name.
 */
public Doge()
{
      this(null);
}

/**
 * Live constructor.
 *
 * Constructor that will call for real name and id of a doge.
 * @param name
 * @param id There isn't a formal parameter id in this method.
 */
/**
 * Constructs this dog with a name.
```

```
     * @param name The name of this dog
     */
    public Doge(String name)
    {
        this.name = name;
        this.id = nextID;
        nextID++;
    }
```