Bluegrass Community and Technical College

Programming Requirements Document

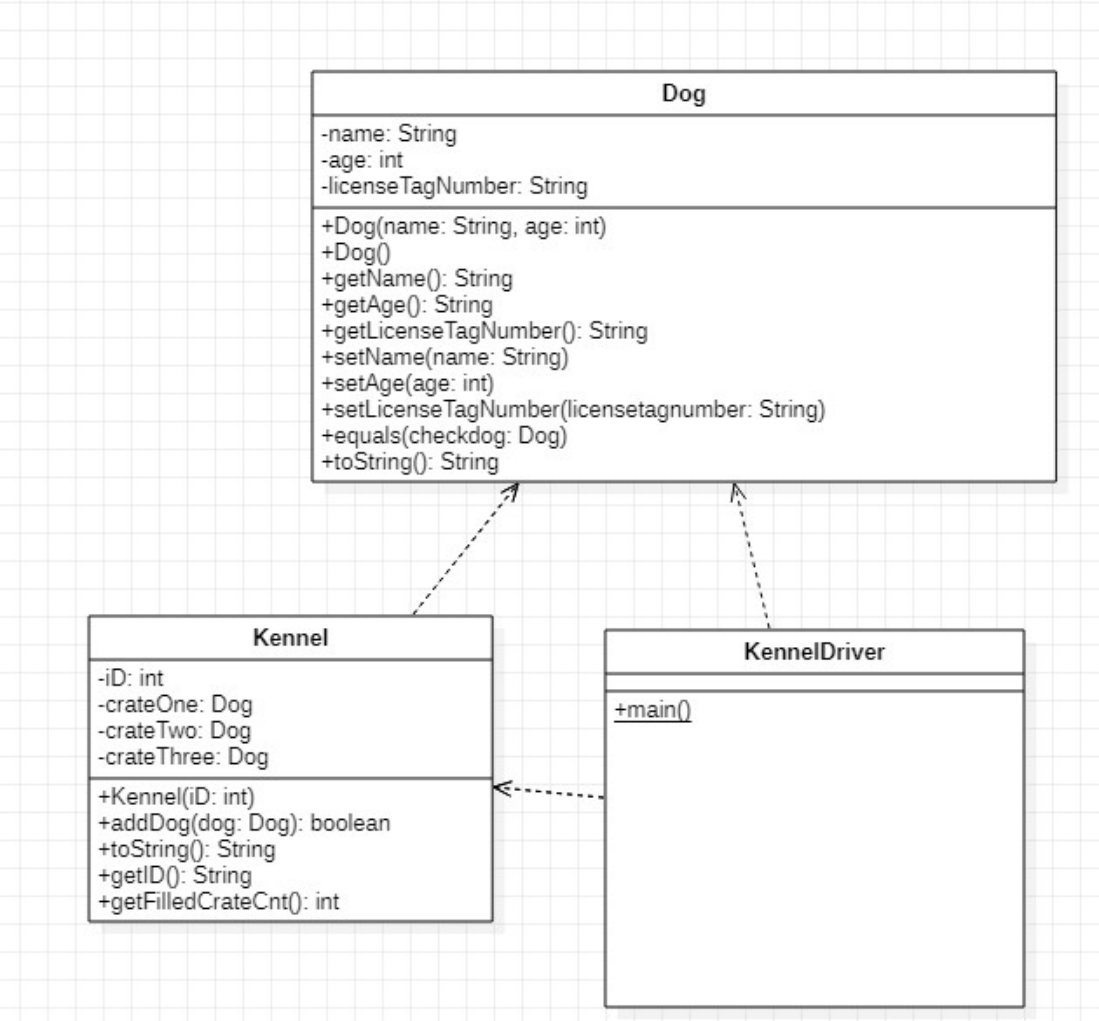
Dogs in a Kennel

NARRATIVE DESCRIPTION

In the last course week we began our study of user-defined Java classes. You were given a program (called a driver) which used a Tile class. You learned how to create a new class in Java by coding the Tile class. This included defining data for the class (color and value), creating two constructors, two accessors, two mutators, an additional method named tradeTiles(), and overriding the toString() method (from the Object class).

This week we will use the concepts to create a program which will help accommodate and place three dogs in a large Kennel in their own big crate.

This is the new UML class diagram for your latest application:



(+ represents the public modifier and – represents the private modifier)

Narrative:

The large dog kennel has four crates used to transport pet animals across small distances in an animal shelter. Each crate can accommodate one dog. **The fourth crate cannot be used because it needs to be used for storage purposes. So only THREE dogs can be accommodated per Kennel.** A program is needed which can take in the data of the dogs and assign them to an available crate in the Kennel in a specific order and print the details of the Kennel and the dogs assigned to the crate once all the crates have been filled.

Kennel Driver Program:

- Create and instantiate the Kennel object.
- Use a while loop and request the user to enter the following details of the Dog needing to be placed in a crate.
  1. Collect the
     - Name of the dog
     - Age of the dog
     - License tag of the dog
  2. Use the collected information to create a object of the type Dog.
  3. Assign the dog object a crate in the Kennel. You can do this by calling the addDog method.
  4. Check if the Kennel is Full and the three crates have been successfully assigned using the getFilledCrateCnt getter method which returns the count of crates which have already been assigned. Your code would look something like

```
public class Kennel
{
    Dog crateOne;
    Dog crateTwo;
    Dog crateThree;

    public int getFilledCrateCnt()
    {
        int cntr = 0;

        if (crateOne != null)
        {
            cntr++;
        }

        if (crateTwo != null)
        {
            cntr++;
        }

        if (crateThree != null)
        {
            cntr++;
        }

        return cntr;

    }
```

5. If all three crates have been assigned, print details of the Kennel using the toString() method with the output looking like below. Feel free to use the toString() method in the Dog class to print the data to the console.

   Kennel Id: 123
   Number of Crates: 03

   CrateOne-Name: Scott
   CrateOne-Age: 3 YEARS
   CrateOne-LicenseTag: 01-1293-222

   CrateTwo-Name: Tiger
   CrateTwo-Age: 2 YEARS
   CrateTwo-LicenseTag: 02-193-222

   CrateThree-Name: Howard
   CrateThree -Age: 5 YEARS
   CrateThree -LicenseTag: 03-193-422

6. Exit the Program.

Constraints:

1. **We can only assign a dog to a crate which has not been assigned. So you will need to check if the crate has been assigned. Your code would look something like below(This is not complete code. You will need to consider the logic validating other two crates also).**

```
public class Kennel
{
    Dog crateOne;
    Dog crateTwo;
    Dog crateThree;

    public  boolean addDog(Dog dog)
    {
        if (dog == null)
        {
            return false;
        }

        if (crateOne == null)
        {
            crateOne = dog;

            return true;
        }
        else
        {
            if (crateOne.equals(dog))
            {
                return false;
            }
        }

        .......

        return false;
    }

}
```

2. **We can only assign one dog to one crate. While assigning a dog to a specific crate, check to see if the same dog has already been assigned to one of the crates of the kennel. You are required to use the equals() method in the Dog class to do that. The equals() method in the Dog class would look something like below.**

```
import java.util.Scanner;

public class Dog
{
    private String licenseTagNumber;

    public String getLicenseTagNumber()
    {
        return licenseTagNumber;
    }

    public boolean equals(Dog dog)
    {
        if (dog == null)
        {
            return false;
        }

        if (this.getLicenseTagNumber().equals(dog.getLicenseTagNumber()))
        {
            return true;
        }

        return false;
    }
```

3. **The while loop should keep continuing until all the three crates in the Kennel are assigned.**

New Concepts Assessed and Illustrated (in addition to any previously learned)

- Instantiating objects from a user-defined class
- Using accessors to retrieve object data
- Adding and using an equals() method to compare two objects
- Using a toString( ) method to display class data

Software Requirements

Standard Requirements listed in previous weeks are implicit requirements now (documentation, free of compile errors, user interface is easy, etc.).

R1: Create working Dog and Kennel classes as per the UML diagram.
R2: Create the Driver program as prescribed above in the "Kennel Driver Program" section of the document.
R2: Create a working equals() method for the Dog class.
R3: Dog objects are clearly displayed.
R4: The user interface is easy to enter the information about the dog needing the crate.
R5: The validations on input data based on constraint are successfully implemented.
R6: Class methods are used correctly.

Security Considerations

The public and private access modifiers have security implications. Use a private modifier for all instance data items. Create accessors for instance data items only when absolutely needed. Do not include sensitive or private information in data returned by the toString() method.