Inheritance & ArrayLists: Pet2 FRQ

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate. Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

The following *Pet* class is used to represent pets and print information about each pet. Each *Pet* object has attributes for the pet's name and species.

```
public class Pet
{
    private String name;
    private String species;

    public Pet(String n, String s)
    {
        name = n;
        species = s;
    }

    public String getName()
    {
        return name;
    }

    public void printPetInfo()
    {
        System.out.print(name + " is a " + species);
    }
}
```

The following *Dog* class is a subclass of the *Pet* class that has one additional attribute: a *String* variable named breed that is used to represent the breed of the dog. The *Dog* class also contains a *printPetInfo* method to print the name and breed of the dog.

```
public class Dog extends Pet
{
    private String breed;

    public Dog(String n, String b)
    {
        super(n, "dog");
        breed = b;
    }

    public void printPetInfo()
    {
        /* to be implemented */
    }
}
```

Consider the following code segment.

```
Dog fluffy = new Dog("Fluffy", "pomeranian");
fluffy.printPetInfo();
```

The code segment is intended to print the following output.

```
Fluffy is a dog of breed pomeranian
```

(a) Complete the *Dog* method *printPetInfo* below. Your implementation should conform to the example above.

```
public void printPetInfo()
```

Consider the following pets.

A rabbit named Floppy
A dog (whose breed is pug) named Arty

The following code segment is intended to represent the two pets described above as objects *pet1* and *pet2*, respectively, and add them to the *ArrayList petList*.

```
ArrayList<Pet> petList = new ArrayList<Pet>();
/* missing code */
petList.add(pet1);
petList.add(pet2);
```

(b) Write a code segment that can be used to replace */* missing code */* so that *pet1* and *pet2* will be correctly created and added to *petList*. Assume that class *Dog* works as intended, regardless of what you wrote in part *(a)*.

The `PetOwner` class below is used to generate a description about a pet and its owner. The `PetOwner` constructor takes a `Pet` object and a `String` value (representing the name of the pet's owner) as parameters.

```
public class PetOwner
{
    private Pet thePet;
    private String owner;

    public PetOwner(Pet p, String o)
    {
        thePet = p;
        owner = o;
    }

    public void printOwnerInfo()
    {
        /* to be implemented */
    }
}
```

Assume that `pet1` and `pet2` were created as specified in part *(b)*. The following table demonstrates the intended behavior of the `PetOwner` class using objects `pet1` and `pet2`.

Code Segment	Result Printed
`PetOwner owner1 = new PetOwner(pet1, "Jerry");` `owner1.printOwnerInfo();`	Floppy is a rabbit owned by Jerry
`PetOwner owner2 = new PetOwner(pet2, "Kris");` `owner2.printOwnerInfo();`	Arty is a dog of breed pug owned by Kris

(c) Complete the `PetOwner` method `printOwnerInfo` below. Your implementation should conform to the examples. Assume that class `Dog` works as intended, regardless of what you wrote in part *(a)*.

```
public void printOwnerInfo()
```