Name _____ Period _____

Pet Store

Your Tasks (Mark these off as you go)

- ☐ Define key vocabulary
- ☐ Interpret the relationship between super and sub classes
- ☐ Create objects using inheritance hierarchies
- ☐ Receive credit for this lab guide

☐ Define key vocabulary

Inheritance (as it applies to Java)

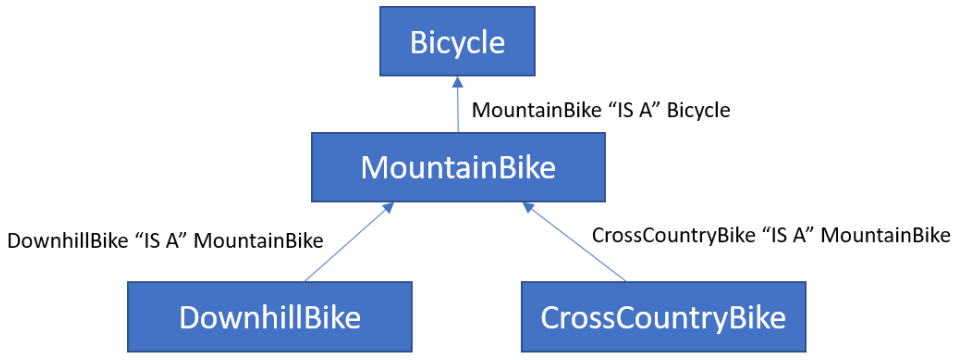
Super class

Sub class

☐ Interpret the relationship between super and sub classes

Recall that inheritance of one class from another follows an "IS A" relationship. That is, a mountain bike "IS A" bicycle. The reverse is not true, however. For example, a bike is not necessarily a mountain bike. When creating objects from super and sub classes, this relationship becomes important.

Consider the following hierarchy of inherited classes between bicycles



The above hierarchy shows the relationship among the classes in a program. According to the hierarchy

- CrossCountryBike "IS A" MountainBike
- DownhillBike "IS A" MountainBike
- MountainBike "IS A" Bicycle

When creating objects from super and sub classes this relationship is enforced. This is illustrated below,

MountainBike “IS A” Bicycle

```
Bicycle myBike = new MountainBike();
```

Bicycle is not necessarily a MountainBike

```
MountainBike myBike = new Bicycle();
```

Another way to think of this is that "parents can make children, but children cannot make parents"

Refer to the Pet, Cat, and Fish classes below,

```
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 String getSpecies(){
            return species;
        }

        public String toString(){
            return getName() + " is a " + getSpecies();
        }
}
```

```
public class Cat extends Pet{
    private String breed;
    public Cat(String n, String b){
        super(n, "Cat");
        breed  = b;
    }

    public void speak(){
        System.out.println("Meow, Meow");
    }

    public String toString(){
        String msg = super.toString() + " of breed " + breed;
        return msg;
    }
}

public class Fish extends Pet{
    private String breed;
    public Fish(String n, String b){
        super(n, "Fish");
        breed  = b;
    }

    public void speak(){
        System.out.println("Blub, Blub");
    }

    public String toString(){
        String msg = super.toString() + " of breed " + breed;
        return msg;
    }
}
```

(a) Draw a diagram to represent the relationship between the Pet, Cat, and Fish classes.

(b) For each of the following (i) Indicate whether the statement is valid (V) or invalid (I) (ii) If the statement is not valid, indicate why.

Statement	V/I	If "I", indicate why.
`Fish f = new Fish("Dory", "Blue Tang");`		
`Cat c = new Fish("Fred", "Siamese");`		
`Fish f = new Pet("Nemo", "Clownfish");`		
`Pet p = new Fish("Dory", "Blue Tang");`		
`Object o = new Cat("Fred", "Ragdoll");`		
`Object o = new Pet("Ravioli");`		

(c) What is the value of s?

```
Pet pet1 = new Pet("Ravioli", "Cat");
String s = pet1.toString();
```

(d) What is printed?

```
Cat pet2 = new Cat("Bean", "Calico");
Pet pet3 = pet2;
pet3.speak();
```

Refer to the code block below to indicate what is printed for each of the following statements.

```
Pet pet1 = new Pet("Princess", "Gorilla");
Cat cat1 = new Cat("Roscoe", "Maine Coon");
Fish fish1 = new Fish("Nemo", "Clownfish");
Pet fish2 = new Fish("Dory", "Blue Tang");
```

(i) `System.out.println(cat1 instanceof Pet);//returns true if cat1 is an instance of Pet`

(ii) `System.out.println(new Cat() instanceof Pet);`

(iii) `System.out.println(pet1);`

(iv)
```
System.out.println(cat1);
```

(v)
```
System.out.println(fish2);
```

(vi)
```
Pet[] fish = new Pet[3];
fish[0] = fish1;
fish[1] = fish2;
fish[0].speak();
```

□ Create objects using inheritance hierarchies

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 String getSpecies(){
            return species;
        }

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

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
```

Complete the Dog method `printPetInfo` below. Your implementation should conform to the example above.

The `PetMaker` class contains the main method for the program. Write code that could be used to create the following pets,

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

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 above in the `PetMaker` class. 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`

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 previously.

☐ Receive credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.