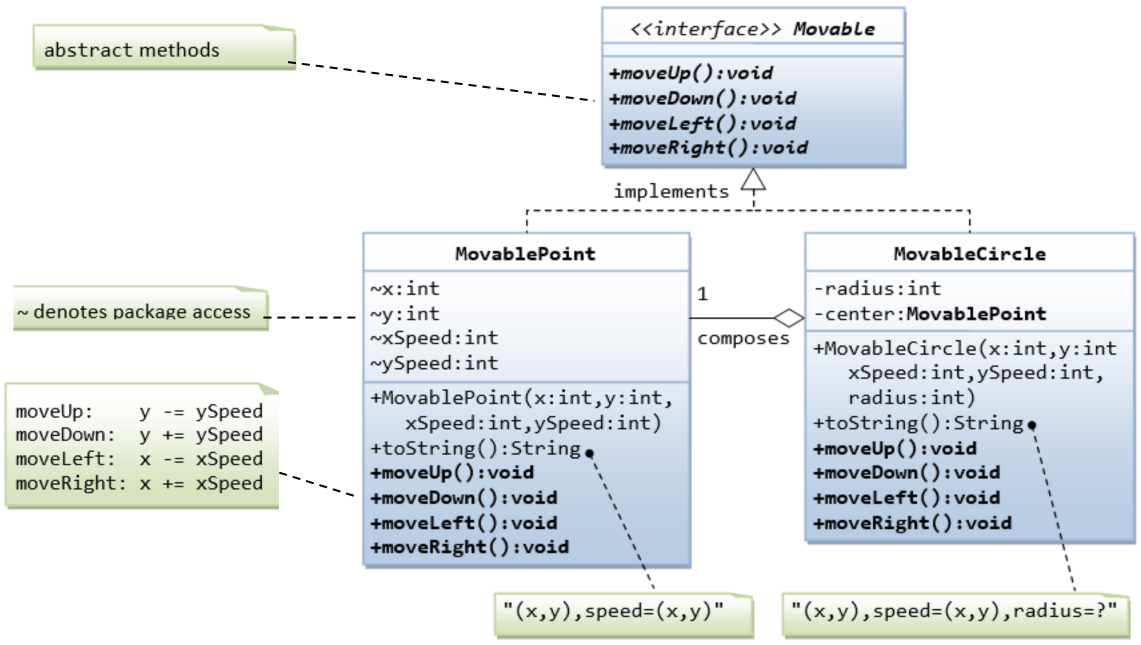
Practice Activities #4

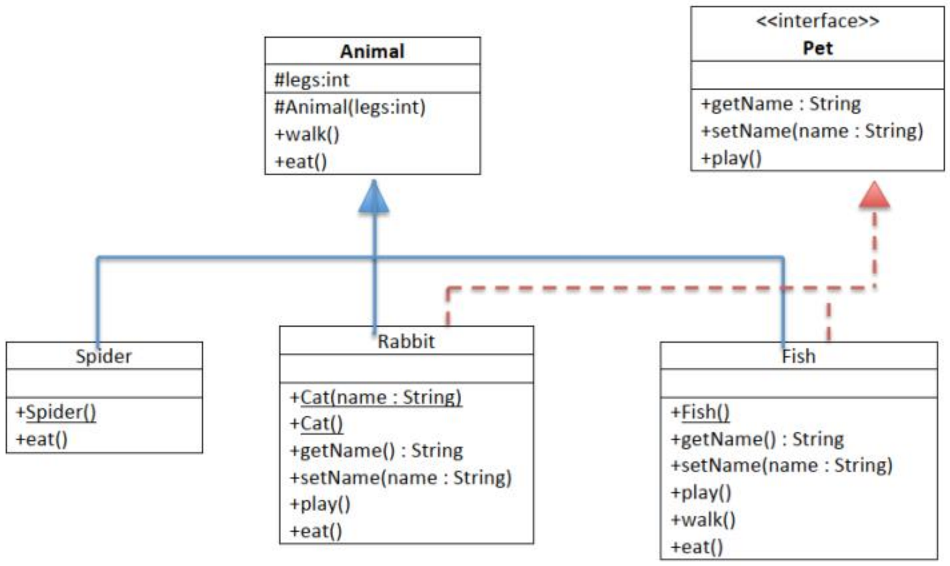
A problem

Write an interface called `Movaable`, which contains
4 `abstract` methods `moveUp()`, `moveDown()`, `moveLeft()` `and` `moveRight()`, as shown in the class diagram. Also write the implementation classes called `MovablePoint` and `MovableCircle`. Mark all the overridden methods with annotation `@Override`.



B problem

In this exercise you will create a hierarchy of animals that is rooted in an abstract class `Animal`. Several of the animal classes will implement an interface called `Pet`. You will experiment with variations of these animals, their methods, and polymorphism.



1. Create the `Animal` class, which is the abstract superclass of all animals.

    - Declare a protected integer attribute called `legs`, which records the number of legs for this animal.
    - Define a protected constructor that initializes the `legs` attribute.
    - Declare an abstract method `eat`.
    - Declare a concrete method `walk` that prints out something about how the animals walks (include the number of legs).

2. Create the `Spider` class.

    - The `Spider` class extends the `Animal` class.
    - Define a default constructor that calls the superclass constructor to specify that all spiders have eight legs.
    - Implement the `eat` method.

3. Create the `Pet` interface specified by the UML diagram.
4. Create the `Cat` class that extends `Animal` and implements `Pet`.

- This class must include a `String` attribute to store the name of the pet.
- Define a constructor that takes one `String` parameter that specifies the cat's name. This constructor must also call the superclass constructor to specify that all cats have four legs.
- Define another constructor that takes no parameters. Have this constructor call the previous constructor (using the `this` keyword) and pass an empty string as the argument.
- Implement the `Pet` interface methods.
- Implement the `eat` method.

5. Create the `Fish` class. Override the Animal methods to specify that fish can't walk and don't have legs.
6. Create an `TestAnimals` program. Have the `main` method create and manipulate instances of the classes you created above. Start with:

```
Fish d = new Fish();
Cat c = new Cat("Fluffy");
Animal a = new Fish();
Animal e = new Spider();
Pet p = new Cat();
```

Experiment by: a) calling the methods in each object, b) casting objects, c) using polymorphism, and d) using `super` to call super class methods.

C problem

Define an interface called `PersonInterface`. This interface containt the following abstract methods.

```
void setName(String myName);

void setAge(int myAge);
```

Define another interface called `SportInterface`. This interface containt the following abstract methods.

```
String getMyFavoriteSport();

void setMyFavoriteSport(String sportName);

int howMuchItCostToPlayThisSport();
```

Define the third interface called `HobbyInterface`. The interface contains the following abstract methods.

```
String whatIsMyHobby();

void setMyHobby(String hobby);
```

Write a class called `SportAndHobbyImpl` that implements the all three interfaces defined above. It sets other fields - `MyFavoriteSport` and `MyHobby` - via `setMyFavoriteSport(String sportName)` and `setMyHobby(String hobby)` methods.

It computes `theCostToPlayThisSport` as following:

age * 10

Display the `Name, Age, MyFavoriteSport, MyHobby, theCostToPlayThisSport` fields