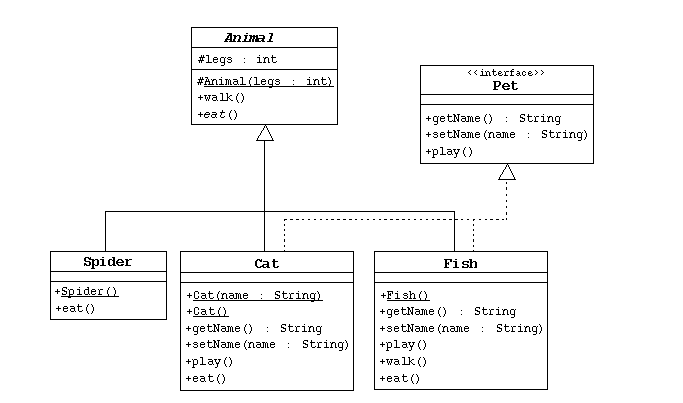
**Objective**

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.
   1. Declare a protected integer attribute called legs, which records the number of legs for this animal.
   2. Define a protected constructor that initializes the legs attribute.
   3. Declare an abstract method eat.
   4. Declare a concrete method walk that prints out something about how the animals walks (include the number of legs).
2. Create the Spider class.
   1. The Spider class extends the Animal class.
   2. Define a default constructor that calls the superclass constructor to specify that all spiders have eight legs.
   3. 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.
   1. This class must include a String attribute to store the name of the pet.
   2. 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.
   3. 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.
   4. Implement the Pet interface methods.
   5. 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:
7. Fish d = new Fish();
8. Cat c = new Cat("Fluffy");
9. Animal a = new Fish();
10. 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.

Q-2)

Write the interface called GeometricObject, which declares two abstract methods: getParimeter() and getArea().

Write the implementation class Circle, with a variable radius, which implements the interface GeometricObject.

Write a test program called TestCircle to test the methods defined in Circle.

Note:

Formula for Perimeter of Circle = 2pr

Formula for Area of Circle = pr2

The class ResizableCircle is defined as a subclass of the class Circle, which also implements an interface called Resizable.

The interface Resizable declares an abstract method resize(), which modifies the dimension (such as radius) by the given percentage.

Write the interface Resizable and the class ResizableCircle.

Write a test program called TestResizableCircle to test the methods defined in ResizableCircle.

Q-3)

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