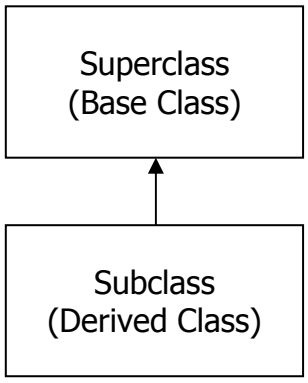
AP Computer Science Name: _____
Karel Chapter 4 Review Notes (6)

POLYMORPHISM

Review of Inheritance:

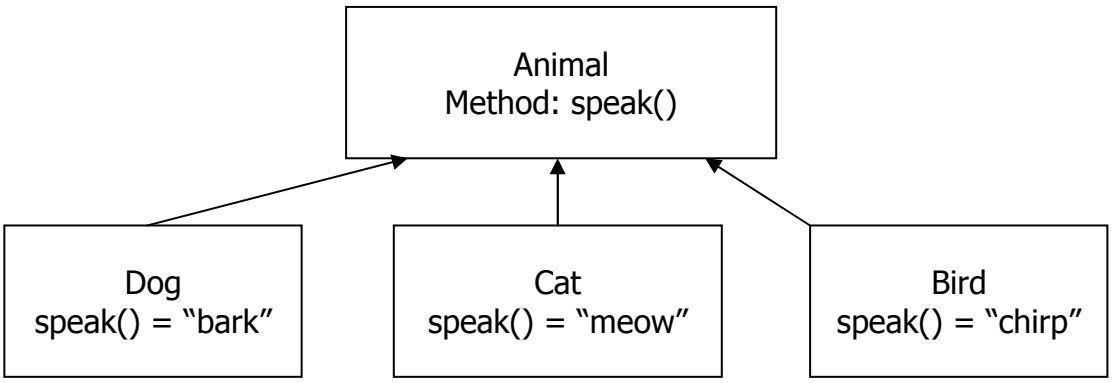
- A class can extend another class, this is called ***inheritance.***
- The base class is called a ***superclass*** and the derived class is called a ***subclass***.
- A subclass inherits all the fields and methods of its superclass (but not its constructors).
- An object of a subclass also inherits the type of the superclass as its own, more generic type.



Inheritance represents the IS-A relationship between types of objects. A superclass defines more general features of the objects of its subclass. A subclass defines additional, more specific features (fields and methods) and may override (redefine) some of the methods of the superclass.

Class Hierarchies:

- Duplicate code in several classes is not only wasteful, but also bad for software maintenance. (If you ever need to change code, you would have to change it everywhere it appears)
- If you define a number of classes for a project and realize that these classes share a lot of code, it may be a good idea to *factor out* the common field and methods into one common superclass.



Polymorphism: Different objects interpreting the same message in a different way.

Abstract Class: A class with some of the methods declared but undefined.

Example:
```
public abstract void speak();
```

*Each subclass must define this method before we can even try to run our program.

If a class has at least one abstract method, it must be declared `abstract`.

Example:
```
public abstract class Animal
{
      …
      public abstract void speak();
      …
}
```

Abstract Classes:

- Java doesn't allow us to create objects of an abstract class
- A class with no abstract methods is called a **concrete class**.

Review of Constructors:

- Every class has at least one constructor. If no constructors are explicitly defined, the compiler provides a default no-args (NO ARGUMENTS) constructor. If at least one constructor is explicitly defined by the programmer, the compiler does not supply a default no-args constructor.
- If a constructor has a call to **`super`**, it must be the first statement in the constructor. The number and types of parameters passed to `super` must match the number and types of parameters expected by one of the constructors of the superclass.
- If `super(…)` does not appear in the constructor of a subclass, then the no-args constructor of the superclass is called by default. In that case, the superclass must have a no-args constructor.

Polymorphism (from Greek: *poly* = many; *morph* = form):

- Ensures that the object's method is called for an object of a specific type, even when that object is disguised as a reference to a more generic type (that is, the type of some ancestor higher up the inheritance line).