抽象类：
An abstract method has no body
If you declare an abstract method, you must mark the class  abstract
But in an abstract class you can mix both the abstract and nonabstract methods
Abstract methods exist solely for polymorphism.
A concrete class must implement all abstract methods.
Implementing an abstract method is just like overriding a method.
eg:

```
puiblic interface Pet{
public abstract void beFriendly();// interface method ares are implicitly public and abstract
public abstract void play();    // have no body
}
```

class from different inherit trees can implements the same interface

```
//: c08:RandVals.java
// Initializing interface fields with
// non-constant initializers.
import java.util.*;
public interface RandVals {
Random rand = new Random();
int randomInt = rand.nextInt(10);
long randomLong = rand.nextLong() * 10;
float randomFloat = rand.nextLong() * 10;
double randomDouble = rand.nextDouble() * 10;
} ///:~
```

在接口中定义的数据成员自动是static 和final 的。它们不能是“空final”，但是可以被非常量表达式初始化。
这些数据成员不是接口的一部分，只是被存储在该接口的静态存储区域内。
接口可以嵌套在类或其它接口中
所以接口中的属性必然是常量，只能读不能改，这样才能为实现接口的对象提供一个统一的属性。你认为是要变化的东西，就放在你自己的实现中，不能放在接口中去，接口只是对一类事物的属性和行为更高层次的抽象。
abstract class cannot be instanced

- All objects come out of an ArrayList<Object> as type Object (meaning, they can be referenced only by an Object reference variable, unless you use a *cast*).
- Multiple inheritance is not allowed in Java, because of the problems associated with the "Deadly Diamond of Death". That means you can extend only one class (i.e. you can have only one immediate superclass).
- An interface is like a 100% pure abstract class. It defines *only* abstract methods.
- Create an interface using the `interface` keyword instead of the word `class`.
- Implement an interface using the keyword `implements`
  Example: `Dog implements Pet`
- Your class can implement multiple interfaces.
- A class that implements an interface *must* implement all the methods of the interface, since ***all interface methods are implicitly public and abstract***.
- To invoke the superclass version of a method from a subclass that's overridden the method, use the `super` keyword. Example: `super.runReport();`

- When you don't want a class to be instantiated (in other words, you don't want anyone to make a new object of that class type) mark the class with the `abstract` keyword.
- An abstract class can have both abstract and non-abstract methods.
- If a class has even *one* abstract method, the class must be marked abstract.
- An abstract method has no body, and the declaration ends with a semicolon (no curly braces).
- All abstract methods must be implemented in the first concrete subclass in the inheritance tree.
- Every class in Java is either a direct or indirect subclass of class **Object** (java.lang. Object).
- Methods can be declared with Object arguments and/or return types.
- You can call methods on an object *only* if the methods are in the class (or interface) used as the *reference* variable type, regardless of the actual *object* type. So, a reference variable of type Object can be used only to call methods defined in class Object, regardless of the type of the object to which the reference refers.
- A reference variable of type Object can't be assigned to any other reference type without a *cast*. A cast can be used to assign a reference variable of one type to a reference variable of a subtype, but at runtime the cast will fail if the object on the heap is NOT of a type compatible with the cast.
  Example: `Dog d = (Dog) x.getObject(aDog);`

First, if you override everything, the base class should really be an interface, not a class. There's no point in implementation inheritance if you don't use any of the inherited methods.

Second, and more importantly, you don't want a stack to support all ArrayList methods.