Lecture 11: Abstract Classes and Interfaces

1 Abstract Classes

An abstract class contains abstract methods (think of these as method-placeholders) that are implemented by concrete subclasses.

Example: Every GeometricObject should provide getArea() and getPerimeter() methods, but unless we know what actual shape it is, these methods cannot be implemented.

Syntax:

```
public/protected abstract ClassName {
    ...
    abstract returnType methodName () ;
    ...
}
```

Reasons for abstract classes:

- provide base/superclass class that guarantees that all subclasses provide certain methods (subclasses have to implement abstract methods of its superclass);
- ability to write more "generic".

2 Interfaces

An interface is a class-like construct that contains only constants and abstract methods.

Syntax:

```
interface Name {
    ...
}
```

Note that all data fields have to be public static final and all methods have to be public abstract (since there is no choice about it, you can omit the access modifiers in definitions of interfaces).

A class can implement multiple interfaces.
2.1 Comparable interface provided by Java

A class that implements Comparable interface has to provide compareTo() method. That is the only requirement of the Comparable interface.

See Loan.java, and SortLoans.java that we wrote in class.