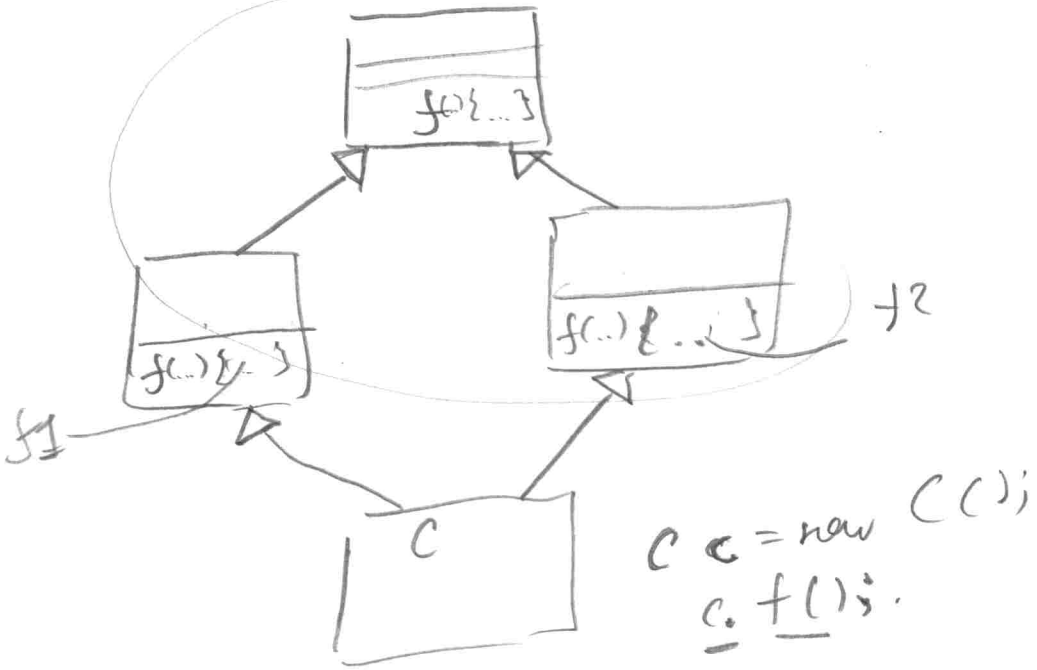
1. Inheritance is an ability to inherit function implementations (behaviors) by a subclass from a superclass

Simula-67

Benefit: software re-use (re-using the effort of writing the code and TESTING IT)

Downside: class hierarchy is done according to a single criterion



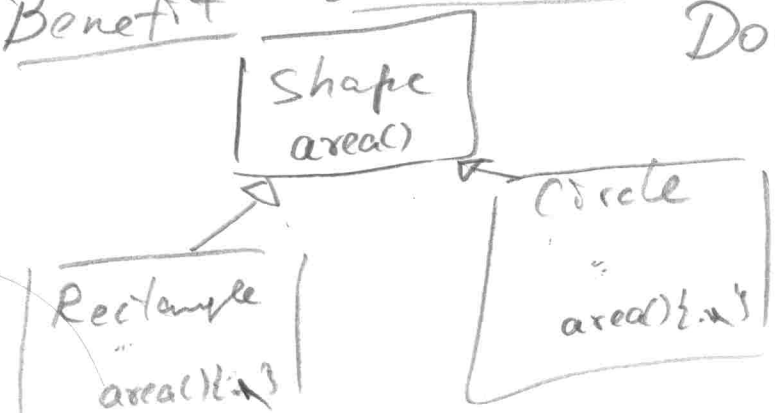
Polymorphism (multiple representations:

an ability to provide multiple implementations for a single function declaration;

a subclass should be able to substitute for its superclass

Benefit: delivers more succinct code

Downside: complicates testing



ArrayList<Shape> shapes = new ArrayList<S

```
[ ...
float sum = 0.0f;
for (Shape shape : shapes) {
    sum = sum + shape.area();
}
```

is assigned

Encapsulation - hiding object's state

```
class Table {
    private HashMap table = new HashMap(
    private int size;
    public void addElement(String key,
                           Object obj){
        table.put(key, obj);
        size++;
    }
```

```
Table table = new Table();
table.size = 10;
```

Benefit: increases reliability

downside: complicates testing

module
- library - module that contains operations
- data pool - module that contains attributes
- object - module that contains RELATED attributes and operations
  - class - an object that has constructor functions

attributes → fields
operations → functions → method (function that belongs to a class)

class - a container for function implementations

type - a hierarchical set can be defined by enumeration or by condition
"enum"

```
interface Singer {
    public void sing();
}

public class SingerAndWriter implements SongWriter {
    Singer, SongWriter {
    sing() {...};
    writeSong() {...};
```

```
interface SongWriter {
    public void writeSong()
}
```