Encapsulation

Hsuan-Tien Lin

Deptartment of CSIE, NTU

OOP Class, March 14, 2013

Recall: Basic OOP Needs

Decompose computation into interactions of "computing parts" called *objects*, each containing its own data (to be manipulated by itself) and own code (to be called by other objects)

- designing object (what variables? what methods?) class
- creating "first" object and calling its first action java ClassName / ClassName.main
- creating other objects new, constructor
- calling other objects method call (this)
- manipulating object status instance var
- deleting objects GC, finalizer

Encapsulation

Decompose computation into interactions of "computing parts" called *objects*, each containing its own data (to be manipulated by itself) and own code (to be called by other objects)

- preventing others from manipulating the object
- letting others call the allowed code

Encapsulation (1/5)

```
1  class Record{
2    String name;
3    String password;
4  }
5
6  public class RecordDemo{
7    public static void main(String[] argv){
8      Record r;
9      String s, p;
10     s = getLoginNameFromUser();
11     r = getRecordFromFile(s);
12     System.out.println(r.password);
13     p = getPasswordFromUser();
14     if (p.equals(r.password)){
15       // ...
16     }
17   }
18 }
```

- if password not encoded, the SYSOP might easily get your password by `getRecordFromFile`

Encapsulation (2/5)

```
1  class Record{
2    String name;
3    String encoded_password;
4  }
5
6  public class RecordDemo{
7    public static void main(String[] argv){
8      Record r;
9      String s, p;
10     s = getLoginNameFromUser();
11     r = getRecordFromFile(s);
12     p = getPasswordFromUser();
13     if (YOUR_ENODING(p).equals(r.encoded_password)){
14       // ...
15     }
16     //A new and careless programmer adds this line
17     r.encoded_password = null;
18   }
19 }
```

- even when password encoded, a careless programmer may make stupid bugs

Encapsulation (3/5)

```
1  class Record{
2    private String encoded_password;
3    public String get_encoded_password(){
4      return encoded_password;
5    }
6  }
7  public class RecordDemo{
8    public static void main(String[] argv){
9      Record r;
10     String s, p;
11     s = getLoginNameFromUser();
12     r = getRecordFromFile(s);
13     p = getPasswordFromUser();
14     if (YOUR_ENODING(p).equals(r.get_encoded_password())){
15       // ...
16     }
17     //A new and careless programmer adds this line
18     r.encoded_password = null; //won't work
19   }
20 }
```

- what if you want to set a new password?

Encapsulation (4/5)

```
1  class Record{
2    String name;
3    private String encoded_password;
4    public String get_encoded_password(){
5      return encoded_password;
6    }
7    public void set_encoded_password(String raw_password){
8      if (blahblah)
9        encoded_password = YOUR_ENCODING(raw_password);
10   }
11 }
```

- **separate implementation and use**: you implement the `Record` class, and other people (possibly you after two years) use it
- **don't trust other people**: silly mistakes can happen
- **hide unnecessary details** (a.k.a. instance variables)
- **think about possible correct/incorrect use of your class**: check them in the methods

Encapsulation (5/5)

```
1  public class RecordDemo{
2    public static void main(String[] argv){
3      Record r;
4      String s, p;
5      s = getLoginNameFromUser();
6      r = getRecordFromFile(s);
7      p = getPasswordFromUser();
8      if (r.match_password(p)){
9        //no need to show encoding to the outside
10     }
11     r.set_encoded_password(old_password, new_password);
12     //don't want this to happen
13     r.encoded_password = null;
14   }
15 }
```

- freedom on making assignments: a potential hole to do bad and/or make bugs

Encapsulation: Key Point

as a designer, you should avoid giving the users of your code too much freedom to do bad and/or make bugs