Softwarekonstruktion - Exercise 2

2 Design Patterns

This exercise should be solved until Friday, October 29th, 2010.

2.1 Design Patterns

2.1.1 How are design patterns described (notation, template)?

2.1.2 MVC Design Pattern

The following cloze texts represent a 'modulo 5 counter' in Java with the MVC design pattern applied. Fill the gaps with the right Java code (regarding to the MVC design pattern), and implement a class containing the `main()`-method. The cloze texts can be downloaded on the lecture's homepage.

```
1  // CounterModel with MVC-MODEL
2  import java.util.*; // Observable
3
4  public class CounterModel extends _____________
5  {
6          private int count = 0;
7
8          public int getCount()
9          {
10                 return count;
11         }
12
13         public void doSomething()
14         {
15                 count = (count + 1) % 5;        // count value changes
16                 setChanged();                          // show changes
17                 notify_____________(new Integer(count));
18         }
19 }
```

Listing 1: CounterModel.java

```
1  // CounterView with MVC-Model
2  import java.util.*; // Observer
3
4  public class CounterView implements Observer
5  {
6          private _____________ model;
7
8          private _____________ controller;
```

```
9
10          public CounterView(_____________ model, String title)
11          {
12                  System.out.println(title);
13                  // Model
14                  this.model = model;
15                  this.model.add____________(this);
16                  // Controller
17                  controller = ____________();
18                  // View
19                  makeView();
20          }
21
22          private CounterController makeController()
23          {
24                  return new ____________(model, this);
25          }
26
27          private void makeView()
28          {
29                  System.out.println( "" + ____________.getCount());
30                  System.out.print( "Continue (y/n): ");
31                  ____________.start();
32          }
33
34          public void release()
35          {
36                  ____________.release();
37                  ____________ = null;
38                  // Model
39                  model.delete____________(this);
40                  model = null;
41          }
42
43          public void update(____________ m, Object o)
44          {
45                  if(model == m)
46                  {
47                          System.out.println( "" + ____________.getCount());
48                          System.out.print( "Continue (y/n): ");
49                  }
50          }
51  }
```

Listing 2: CounterView.java

```
1  // CounterController.java with MVC-Model
2  import java.io.*; // Keyboard input
3
4  public class CounterController
5  {
6          private ____________ model;
7
8          private ____________ view;
9
10         public CounterController(____________ model, ____________ view)
11         {
12                 this.model = model;
13                 this.view = view;
14         }
15
```

```
16          public void release()
17          {
18                  model = null;
19                  view = null;
20          }
21
22          public void start()
23          {
24                  String s = "";
25                  BufferedReader br = new BufferedReader (new InputStreamReader(
                        System.in));
26                  do
27                  {
28                          try {s = br.readLine();} catch (IOException e) {System.
                                out.println("readLine() error");};
29                          if(s.equals("y")) _______.doSomething();
30                          if(s.equals("n")) break;
31                  }
32                  while(true);
33                  try {br.close();} catch (IOException e) {System.out.println("
                        close() error");};
34                  _______.release();
35          }
36  }
```

Listing 3: CounterController.java

2.1.3 Behavior of MVC Design Pattern

Draw a UML sequence diagram representing the behavior of the 'modulo 5 counter' developed in exercise 2.1.2.