Design-pattern

Når bruke hvilket design-pattern:

- **Model, View, Controller(MVC)**
  Større programmer hvor det blir rotete å ha GUI, data og funksjonalitet sammen.
- **Singleton**
  Når du skal ha et objekt som hele programmet skal kunne bruke.
- **Proxy**
  Når du har noe som går trengt, og du trenger å gjøre det en gang og mellomlagre resultatet.
- **Adapter**
  Når du vil lagre et enklere grensesnitt for en annen klasse.

Eksempler:

```
1  class SimpleArray{
2     int i = 0
3     int[] a = new int[100;]
4     public void push(value){
5        a[i] = value;
6        i++;}
7     public void pop(){
8        i--
9        return a[i];
10       }}
```

Adapter er riktig

```
1  class GetUsernames{
2     private static GetUsernames u = new GetUsernames();
3     public static GetUsernames getInstance(){
4        return u;
5     }}
```

Singleton er riktig

```
1  class Downloader{
2     HashMap<String,String> images = new HashMap<String,String>();
3     public void getImage(URL){
4        Image i = images.get(URL);
5        if(i!=null){return i;}
6        images.put(URL,download(URL));
7        return images.get(URL);
8        }
9  }
```

Proxy er riktig