Exercices de Patterns (3)

1) Implémentation

Soient les classes DVD, Livre, MagasinDeDVD et MagasinDeLivre

a) Refactorez le code afin qu'il implémente le pattern Factory Method. Ecrivez un main qui instancie un magasin de DVD et y ajoute 3 DVDs, puis qui instancie un magasin de Livre et y ajoute 4 livres.

b) Dans un autre projet, refactorez le code afin qu'il implémente le pattern Abstract Factory. Ecrivez un main qui instancie un magasin de DVD et y ajoute 4 DVDs, puis qui instancie un magasin de Livre et y ajoute 3 livres.

2) Reconnaissance (question de septembre 2016)

Pour écrire le code source du langage Java, les développeurs ont utilisé intensivement les patterns. Dans les deux codes proposés suivants issus des librairies java, identifiez un pattern utilisé. Donnez la correspondance entre les noms des classes utilisées dans la théorie et celle du code.

Pour plus d'informations à propos de ces codes, n'hésitez pas à consulter l'API Java. Le code suivant provient de la classe java.lang.Integer

a)

```
public static Integer valueOf(int i) {
          final int offset = 128;
          if (i >= -128 && i <= 127) { // must cache
                return IntegerCache.cache[i + offset];
          }
          return new Integer(i);
    }

    private static class IntegerCache {
          private IntegerCache() {
          }

          static final Integer cache[] = new Integer[-(-128) + 127 + 1];

          static {
                for (int i = 0; i < cache.length; i++)
                      cache[i] = new Integer(i - 128);
          }
    }
```

<u>Nom du Pattern</u> :

Participants théoriques **Noms utilisés dans le code**

Nom théorique des méthodes **Nom utilisé dans le code**

b) Le code suivant provient des classes java.io.Reader et java.io.BufferedReader

```
public abstract class Reader implements Readable, Closeable {

    public int read() throws IOException {
           char cb[] = new char[1];
           if (read(cb, 0, 1) == -1)
                  return -1;
           else
                  return cb[0];
    }

    abstract public int read(char cbuf[], int off, int len) throws IOException;
}

public class BufferedReader extends Reader {
    public int read(char cbuf[], int off, int len) throws IOException {
           synchronized (lock) {
                  ensureOpen();
                  if ((off < 0) || (off > cbuf.length) || (len < 0)
                                || ((off + len) > cbuf.length) || ((off + len) < 0)) {
                         throw new IndexOutOfBoundsException();
                  } else if (len == 0) {
                         return 0;
                  }

                  int n = read1(cbuf, off, len);
                  if (n <= 0)
                         return n;
                  while ((n < len) && in.ready()) {
                         int n1 = read1(cbuf, off + n, len - n);
                         if (n1 <= 0)
                                break;
                         n += n1;
                  }
                  return n;
           }
    }
}
```

<u>Nom du Pattern</u> :

Participants théoriques **Noms utilisés dans le code**

Nom théorique des méthodes **Nom utilisé dans le code**