Lab VII.

Objetivos

Os objetivos deste trabalho são:

- Identificar e utilizar padrões relacionados com a estrutura de objetos e classes
- Aplicar boas práticas de programação por padrões em casos práticos

Nota: Para além do código no codeUA, inclua também um ficheiro PDF ou PNG com o diagrama de classes da solução final (pode usar o UMLet, por exemplo, ou um plugin Eclipse/Netbeans).

VII.1 Atribuição dinâmica de responsabilidades

A empresa *TodosFazem* (TF) pretende fazer uma gestão dinâmica de funcionários de forma a poder atribuir responsabilidades diversas ao longo do ano.

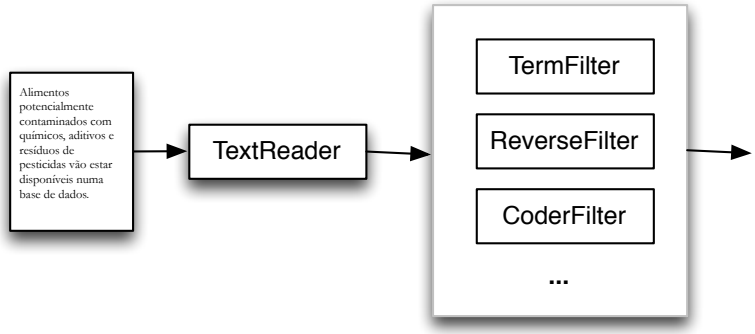
a) Usando como base as entidades seguintes, criando outras se necessário, construa uma solução que permita à empresa gerir empregados e poder atribuir dinamicamente, a cada um, competências distintas. Note que um empregado pode, eventualmente, ter ao mesmo tempo várias competências, por exemplo ser TeamMember e TeamLeader.

b) Crie um programa principal para testar a solução. Por simplicidade pode implementar os métodos apenas com mensagens na consola.

```
Employee
        start (Date)
        terminate(Date)
        work()
TeamMember
        start (Date)
        terminate(Date)
        work()
        colaborate()
TeamLeader
        start (Date)
        terminate(Date)
        work()
        plan()
Manager
        start (Date)
        terminate(Date)
        work()
        manage()
```

VII.2 Processador de texto

Construa uma solução geral que permita ler documentos de qualquer formato (mas na implementação restrinja a ficheiros TXT). O programa deverá permitir ler texto e aplicar um ou mais filtros sobre esse texto.
Tome como base as seguintes entidades, privilegiando a modulação do problema e só depois a implementação de funcionalidades:



- TextReader – lê um ficheiro. Inclui os métodos:
  - boolean hasNext()
  - String next() – devolve parágrafo. Por exemplo:
    Alimentos potencialmente contaminados com químicos, aditivos e resíduos de pesticidas vão estar disponíveis numa base de dados.
- TermFilter – Separa em palavras. Inclui os métodos:
  - boolean hasNext()
  - String next() – devolve termo (por exemplo: *Alimentos*)
- ReverseFilter – inverte a entrada. Inclui os métodos:
  - boolean hasNext()
  - String next() – devolve texto invertido (por exemplo: *sotnemilA*)
- CoderFilter – cifra a entrada (usando uma técnica simples). Inclui os métodos:
  - boolean hasNext()
  - String next() – devolve texto cifrado (por exemplo: *Bl3m2nt4s*)

Exemplos de utilização:

```
...
reader = new TextReader("someFileName");
reader = new CoderFilter(new TextReader("someFileName"));
reader = new ReverseFilter(new TermFilter(new TextReader("someFileName"));
...
```

VII.3 Composição gráfica

Construa uma solução que permita criar as seguintes figuras geométricas:

- Quadrado
- Rectângulo
- Círculo
- Bloco (que é um agregado de figuras geométricas)

Use o programa seguinte para testar a solução:

```
public class GeometricFigures {
    public static void main(String[] args) {
        Bloco principal = new Bloco("Main");
        Bloco top = new Bloco("Top");
        Bloco bot = new Bloco("Bottom");
        top.add(new Rectangulo("jogo"));
        principal.add(top);
        principal.add(bot);
        bot.add(new Circulo("rosa"));
        bot.add(new Quadrado("verde"));
```

```
            top.add(new Bloco("Outra área"));
            principal.draw();
        }
    }
```

Output possível:

```
Window Main
   Window Top
      Rectangulo jogo
      Window Outra área
   Window Bottom
      Circulo rosa
      Quadrado verde
```