Disciplina: Programação Orientada À Objetos
Professor: Dirson
Alunos: Ana Luísa de Bastos Chagas(202200491),
Pedro Lemes Sixel Lobo(202200550)

1 - Sobre padrões de Projeto (Design Patterns) responda as seguintes questões:

a) **Quais as principais caraterísticas dos três tipos de padrões de Projeto (Design Patterns) em Orientação a Objetos do GoF?**

Padrões de Criação: Exigem tratamento em relação à criação/inicialização de objetos, de modo a atender às necessidades variadas que possam existir. Além disso, permitem uma maior flexibilidade e reutilização do código existente.
Padrões Estruturais: Este design pattern está relacionado à composição de objetos e classes, descrevendo itens como elaboração, organização ou ainda a associação entre objetos, classes, interfaces. Ele facilita mudanças no código, sem ter que alterar muitas partes do código fonte ao criar novas classes, alterando o comportamento dos objetos, permitindo a junção de objetos em estruturas mais complexas.

Padrões Comportamentais: Atuam descrevendo o processo de comunicação entre os objetos/classes, gerenciando algoritmos, responsabilidades e relacionamento entre eles. Buscam um “baixo acoplamento”, em geral.

b) **Dê um exemplo de um deles e quando é útil ou apropriado usar este determinado tipo de padrão diferente do que foi exposto em sala e do trabalho final da disciplina (caso o tema seja padrões de projeto).**

Padrões Estruturais: Decorator - O padrão decorator, também chamado de padrão “Wrapper”, permite o acoplamento dinâmico de novos “comportamentos” a certos objetos, sem o alteramento de suas estruturas básicas. Ele funciona pela criação de um conjunto de classes (os decorators/decoradores) que envolvem um objeto base, com cada decorador adicionando uma funcionalidade específica ao objeto. É útil usá-lo quando se tem um objeto base e deseja-se adicionar decorators sem a necessidade de criar subclasses separadas para cada combinação possível de funcionalidades. Isso acontece em, por exemplo, um sistema de geração de relatórios: cria-se a classe base do relatório e usam-se decorators para complementação do projeto (um que adiciona gŕaficos, outro que adiciona tabelas de tendências de mercado, etc., conforme necessário), de modo a obter um sistema completo e flexível.

c) **É possível fazer uma variação deste tipo de padrão escolhido na letra "b"? (Justifique a sua resposta).**

Sim, é possível fazer variações no padrão Decorator, visto que ele é um padrão flexível e capaz de atender a diferentes necessidades. Uma variação (chamada variação hierárquica) dele ocorre nos casos em que há uma dada hierarquia entre seus objetos. Faz-se o uso de aplicação de funcionalidade em camadas a um objeto.

2- **Sobre persistência de dados responda a letra "a" para Arquivos ou a letra "b" para Banco de Dados não ambas.**

Questão escolhida - Banco de dados (letra b):

b)

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  public class conexao1 {
4      private String url;
5      private String usuario;
6      private String senha;
7      private Connection con;
8      conexao1() {
9          url = "jdbc:postgresql://localhost:5432/postgres";
10         usuario = "postgres";
11         senha = "post";
12     try {
13       Class.forName("org.postgresql.Driver");
14       con = DriverManager.getConnection(url, usuario, senha);
15       System.out.println("Qual a menssagem seria apropriada?");
16       } catch (Exception e) {
17          e.printStackTrace();
18       }
19     }
20     public static void main (String[] args){
21             conexao con = new conexao();
22      }
23 }
```

Comentário do código abaixo:

1. Importa do pacote java.sql a classe Connection
2. Importa do pacote java.sql a classe DriverManager

3. Declara a classe pública “conexao1”
4. Declara string privada “url”
5. Declara string privada “usuario”
6. Declara string privada “senha”
7. Declara variável de instância Connection privada “con” do pacote java.sql
8. Método Construtor da classe conexao1 criada antes
9. “url” recebe endereço de acesso, porta e nome do banco de dados
10. “usuario” recebe nome de usuário do banco de dados
11. “senha” recebe senha do banco de dados
12. O try está sendo usado para caso tenha alguma exceção
13. O driver JDBC de conexão com o banco de dados PostgreeSQL está sendo carregado
14. Aqui é feita a conexão com o banco de dados com as informações já obtidas
15. Print no terminal perguntando qual mensagem seria apropriada
16. Exceção para caso ocorra erro ao tentar conectar com o banco de dados
17. Impressão da mensagem de erro na tela
18. - Fechamento de chave
19. - Fechamento de chave
20. Método público e estático “main”, responsável por executar o programa
21. Cria objeto conexão para invocar o construtor
22. - Fechamento de chave
23. - Fechamento de chave

Na linha 15, onde há um System.out.println(“Qual seria a mensagem apropriada”), a mensagem ideal seria uma que avisasse que a conexão com o banco de dados deu certo, como “Conexão feita com sucesso”, por exemplo.

Como a linha 17 ocorrer caso tenha um problema/exceção, tem que haver um problema no bloco try que impeça a conexão com o banco de dados.

3 - **Sobre a ferramenta Junit 5 (Java) ou alternativamente Unit Test ou Pytest (Python) responda a seguinte questão. Uma das características mais marcantes da Orientação a Objetos é o reuso de software e para reusar um software desenvolvido e necessário testá-lo. Escolha um código-fonte qualquer e faça uma classe de teste usando a ferramenta Junit 5 (Java) ou alternativamente Unit Test ou Pytest (Python), ou seja, nesta classe de teste não utilize o método main e sim a ferramenta de teste da linguagem. Explique o que está sendo testado como comentário inserido na própria classe de teste? Obs.: Envie todo o código-fonte, inclusive a classe de teste zipado como um arquivo de nome questao3_PrimeiroNomedoEstudande(s).zip.**

(A resposta do item 3 está enviada no arquivo de código)

4 - **Uma das formas de usar Threads em Orientação a Objetos é para implementar a concorrência. Abaixo temos dois links com exemplos de concorrência em Java e em Python respectivamente. Sobre este assunto responda as seguintes questões.**
a) Copie o código-fonte utilizado e explique o funcionamento da concorrência de um exemplo do link em Java ou Python.
public class SimpleThreads {

Segue o código fonte em java:

```
  // Display a message, preceded by
  // the name of the current thread
  static void threadMessage(String message) {
      String threadName =
          Thread.currentThread().getName();
      System.out.format("%s: %s%n",
                        threadName,
                        message);
  }

  private static class MessageLoop
      implements Runnable {
      public void run() {
          String importantInfo[] = {
              "Mares eat oats",
              "Does eat oats",
              "Little lambs eat ivy",
              "A kid will eat ivy too"
          };
          try {
              for (int i = 0;
                   i < importantInfo.length;
                   i++) {
                  // Pause for 4 seconds
                  Thread.sleep(4000);
                  // Print a message
                  threadMessage(importantInfo[i]);
              }
          } catch (InterruptedException e) {
              threadMessage("I wasn't done!");
          }
      }
  }
```

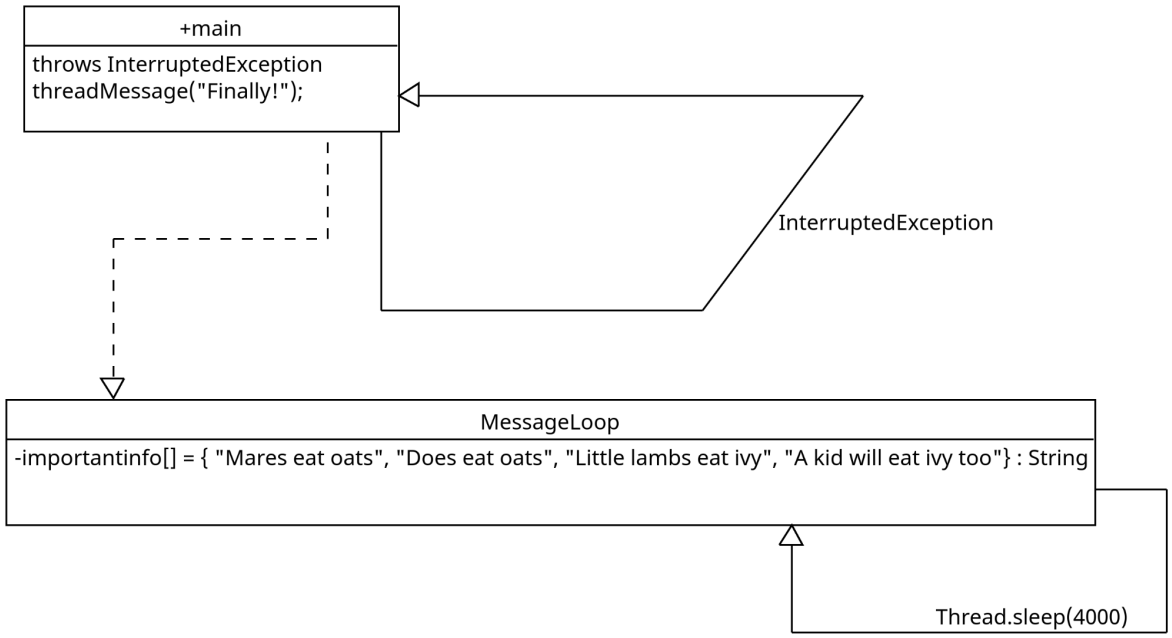
```
public static void main(String args[])
    throws InterruptedException {

    // Delay, in milliseconds before
    // we interrupt MessageLoop
    // thread (default one hour).
    long patience = 1000 * 60 * 60;

    // If command line argument
    // present, gives patience
    // in seconds.
    if (args.length > 0) {
        try {
            patience = Long.parseLong(args[0]) * 1000;
        } catch (NumberFormatException e) {
            System.err.println("Argument must be an integer.");
            System.exit(1);
        }
    }

    threadMessage("Starting MessageLoop thread");
    long startTime = System.currentTimeMillis();
    Thread t = new Thread(new MessageLoop());
    t.start();

    threadMessage("Waiting for MessageLoop thread to finish");
    // loop until MessageLoop
    // thread exits
    while (t.isAlive()) {
        threadMessage("Still waiting...");
        // Wait maximum of 1 second
        // for MessageLoop thread
        // to finish.
        t.join(1000);
        if (((System.currentTimeMillis() - startTime) > patience)
              && t.isAlive()) {
            threadMessage("Tired of waiting!");
            t.interrupt();
            // Shouldn't be long now
            // -- wait indefinitely
            t.join();
        }
    }
    threadMessage("Finally!");
```

```
    }
}
```

No programa acima, a concorrência é usada para realizar múltiplas tarefas ao mesmo tempo, e controlar o fluxo de execução entre as threads criadas. É criada uma thread secundária de nome “message loop”, que exibe ciclicamente mensagens, com tempo de pausa entre elas. A main, como thread principal do programa, aguarda a finalização da thread secundária (message loop) ou espera até que o tempo máximo definido seja excedido.

b) Gere um diagrama de classe com a Ferramenta UMLet do exemplo escolhido na letra “a” no formato padrão da ferramenta (.uxf).

Segue abaixo uma imagem do diagrama (porém, dentre os arquivos zipados, também estará o uxf e um pdf do diagrama, além do código fonte).



5 - **Qual o objetivo do TFD (Trabalho Final da Disciplina) do seu grupo e resuma a sua funcionalidade.**

O objetivo do Trabalho Final da Disciplina do nosso grupo, tendo sido escolhido o tema “Aplicação de Design Pattern - GoF (Padrões de Criação, Estruturais ou Comportamentais)” dentre os oferecidos, é a construção de um sistema de cadastro de usuários utilizando o design pattern de criação “Builder”. Em termos de funcionalidade, ao rodar o código, será exibido um menu colorido via terminal, com

as opções “Sair” (para encerrar a execução), “Criar PF” (com as opções de preenchimento: nome, email, login, senha e CPF), para iniciar a criação de um usuário do tipo pessoa física, “Criar PJ”(com as opções de preenchimento: nome, email, login, senha, CNPJ e tipo de CNPJ), para iniciar a criação de um usuário do tipo pessoa jurídica/empresa, além de duas opções de exibição (Mostrar PF e Mostrar PJ) para facilitar a verificação do programa, de modo a constatar se o salvamento foi feito corretamente. Finalmente, há também a opção “Salvar Dados”, que gera o salvamento dos usuários inscritos em um arquivo de texto. É notável que, visando manter a coerência com a ideia do padrão builder, e oferecer maior conforto ao usuário (não o forçando a inserir desde o início dados que não sejam necessários para a mera criação de uma conta), é obrigatório o preenchimento apenas dos campos “Login” e “Senha”, em conformidade com diversos sites que permitem um cadastro inicial simples como tal - porém, claramente, o usuário pode preencher tudo, ou apenas parte além de “Login” e “Senha” se assim desejar.