UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
SCC0204/SCC0504 – Programação Orientada a Objetos
Prof. Jose Fernando Rodrigues Jr – 1º./2025

Simulado – 2ª. avaliação

Em todos os exercícios:

- só defina o construtor se ele receber algum argumento para criação
- assuma que gets e sets estão definidos, não precisa escrevê-los
- System.out.println("uma String") pode ser substituído por sout("uma String")

(2.0) 1. Padrão de projeto
Implemente o padrão de projeto Chain of Responsibility para simular um sistema de autorização de compras em uma empresa. Considere que uma solicitação de compra possui o valor da compra e uma descrição. Crie a classe SolicitacaoCompra com esses dois atributos. Em seguida, modele uma cadeia de aprovadores por meio da classe abstrata Aprovador, que possui o atributo proximo (do tipo Aprovador) e o método void aprovar(SolicitacaoCompra s), que deve ser sobrescrito pelas subclasses. Crie três subclasses concretas: Supervisor, que aprova compras até R$ 1.000; Gerente, que aprova compras até R$ 10.000; e Diretor, que aprova qualquer valor. Cada classe deve imprimir no console a aprovação ou encaminhar para o próximo na cadeia, se o valor for superior ao seu limite. No método main, crie a cadeia Supervisor → Gerente → Diretor, e envie duas solicitações de compra.

(2.0) 2. Padrão de projeto
Implemente o padrão Observer para simular um sistema de monitoramento de temperatura. Crie a interface ObservadorTemperatura, que define o método void atualizar(double temperatura). Em seguida, crie duas classes que implementam essa interface: PainelPrincipal, que exibe a temperatura no console, e AlarmeTemperatura, que imprime um alerta quando a temperatura ultrapassar 40 graus Celsius. Crie a classe SensorTemperatura, que possui um atributo do tipo lista de observadores. Ao chamar o método setTemperatura(double temp), todos os observadores registrados devem ser notificados. No método main, registre ambos os observadores no sensor, e simule a notificação de dois valores de temperatura.

(2.0) 3. Generics
Implemente uma estrutura de dados do tipo pilha genérica, denominada Pilha<T>. Essa classe deve permitir operações para empilhar (push(T elem)), desempilhar (T pop()), e verificar se está vazia (boolean estaVazia()). No método main, demonstre o uso da pilha criando um exemplo com valores do tipo Integer, realizando pelo menos uma operação de empilhar e uma de desempilhar.

(2.0) 4. Threads
Implemente uma classe chamada LeitorArquivo que implementa a interface Runnable. O construtor da classe deve receber o nome de um arquivo de texto, e seu método run() deve abrir esse arquivo e imprimir no console o número de linhas contidas nele. No método main, crie e inicie duas threads para leitura dos arquivos a.txt, e b.txt, e utilize o método join() para aguardar que todas as threads finalizem.
Código para contar as linhas de um arquivo:

```
try (BufferedReader br = new BufferedReader(new FileReader(nomeArquivo))) {
   while (br.readLine() != null) total++;
} catch (IOException e) {
   sout("Erro ao ler " + nomeArquivo);
}
```

(2.0) 5. Exceções
Implemente uma exceção denominada SaldoInsuficienteException, que deve ser usada em operações de saque em uma conta bancária. Crie a classe Conta, com os atributos titular e saldo, além dos métodos sacar(double valor) e depositar(double valor). O método sacar deve lançar a exceção SaldoInsuficienteException caso o valor a ser sacado seja maior que o saldo disponível. No método main, crie uma conta com R$ 500, tente sacar R$ 800, capture e trate a exceção exibindo a mensagem apropriada.