Exercícios de Herança, Polimorfismo e Interfaces

Animal:

Crie uma interface Animal que define o método *comunicar*. Depois, crie classes para representar humanos, gatos e cachorros. Cada classe deve implementar o método *comunicar*, que exibe uma mensagem na tela, por exemplo, “olá” para humanos e “miau” para gatos.
Realize testes em seu método *main* que animais sejam instanciados e o método comunicar seja chamado para todos.

Conta Corrente:

Crie uma classe para representar uma conta corrente, com métodos para depositar uma quantia, sacar uma quantia, transferir uma quantia para outra conta e obter o saldo. Para cada saque será debitada também uma taxa de operação equivalente a 0,5% do valor sacado. Crie, em seguida, uma subclasse dessa classe anterior para representar uma conta corrente de um cliente especial. Clientes especiais pagam taxas de operação de apenas 0,1% do valor sacado. Crie uma lista de contas para realizar os testes, na qual sejam realizadas operações de saque em todas as contas e exiba o saldo da conta após o saque. No caso de transferências, aplique metade da taxa indicada para saque, ou seja, ao realizar a transferência deve ser debitado 0,25% (conta corrente simples) ou 0,05% (conta corrente de cliente especial) do valor transferido. Realize também algumas transferências entre contas, verificando os saldos após as operações.

Área e Perímetro:

Implemente o seguinte cenário:

a) Uma interface para representar qualquer forma geométrica, definindo métodos que retornam o perímetro e a área da forma;
b) Uma classe para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados;
c) Classes para representar retângulos e quadrados. A primeira deve receber o tamanho da base e da altura no construtor, enquanto a segunda deve receber apenas o tamanho do lado;
d) Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.
e) Implemente o método toString para as classes definidas. O método deve retornar uma String contendo o valor dos lados (ou raio, no caso do círculo) da forma geométrica.

Realize testes deste cenário utilizando uma lista de formas geométricas. Deve ser chamado o toString de todas as formas, além de suas áreas e perímetros.

Exercícios de Arquivos

Observação:

- Durante a aula de arquivos foi utilizado o BufferedWriter e o FileWriter para a escrita de arquivo, e o BufferedReader e o FileReader para a leitura dos arquivos. Estes são um entre diversos outros tipos de classes que permitem a leitura e escrita de arquivos.
- O openCsv foi adicionado no “pom.xml” abaixo das dependências, o arquivo ficaria assim:

```
<dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.opencsv</groupId>
      <artifactId>opencsv</artifactId>
      <version>5.6</version>
    </dependency>
 </dependencies>
```

- Lembrando que no seu “pom.xml” já possui o junit, então não é necessário a adição dele.

Relatório de Alunos:

Supondo uma lista de pelo menos 3 alunos, com: id; nome; data de matrícula; e nota final. Crie um arquivo de relatório contendo estes dados.

Avaliando Alunos:

Supondo o arquivo gerado no exercício anterior, apresente uma lista dos alunos contidos dentro do arquivo no terminal.