Exercício 1

Crie uma classe ContaBancaria que possui um saldo como field e os métodos Sacar(double), Depositar(double) e Transferir(double, ContaBancaria). Crie também duas exceções: ValorInvalidoException e SaldoInsuficienteException.

A exceção ValorInvalidoException deve ser lançada se o valor utilizado nas operações de depósito, saque ou transferência for igual ou inferior a 0. Já a exceção SaldoInsuficienteException deve ser lançada se o valor de um saque ou transferência for superior ao saldo disponível. No construtor de ValorInvalidoException é necessário fornecer uma mensagem de erro e o valor inválido utilizado. E no construtor de SaldoInsuficienteException é necessário fornecer uma mensagem de erro e também o saldo disponível.

Crie uma classe que instancia duas contas e tenta realizar operações de depósito, saque e transferência. Faça transações corretas e também transações que geram exceção. Quando a transação gerar exceção, faça um catch da mesma, imprima a mensagem de erro e o valor inválido utilizado (para ValorInvalidoException) ou o saldo disponível (para SaldoInsuficienteException).

Exercício 2

Imagine que a sua aplicação é composta pelo seguinte código:

```
object o = null;
o.toString();
```

Se você executar este código irá perceber que uma exceção será lançada. Identifique qual a exceção e transforme ela para uma mensagem amigável.

Exercício 3

Crie um *enum* chamado `Exercicio` que pode assumir as opções `Academia`, `Luta` e `Corrida`, com os respectivos valores 1, 2 e 3 associados.

Mostre no console as opções de exercícios existentes no *enum* e solicite ao usuário a digitação, via console, de um exercício (1, 2 ou 3). Mostre então o nome do exercício associada à opção digitada.

Dica: Se você desejar converter a `string` retornada por `Console.ReadLine()` em um `int`, você pode usar o método `int.Parse()` e fornecer a `string` como parâmetro. Caso a conversão não possa ser realizada, esta chamada vai lançar uma exceção do tipo `FormatException`.

Exercício 4

Crie uma classe chamada `ServicoFabrica<T>` que possui um método chamado `NovaInstancia()`. Quando este método é chamado, ele cria um objeto do tipo `T` (invocando o construtor padrão, sem parâmetros) e retorna este objeto. Um detalhe importante é que apenas classes que implementam a interface `IServico` (que também deve ser definida por você) devem ser aceitos na parametrização do tipo. Esta interface possui o método `void Executar()`, que deve ser implementado pelas classes que implementam esta interface.

Exercício 5

Um triângulo é uma figura geométrica que possui três pontos. Crie as estruturas `Triangulo` e `Ponto` para representar estes conceitos.

A estrutura `Ponto` possui as properties `X`, `Y` e `Z`, que correspondem às coordenadas dos pontos, e tipo de dado das coordenadas deve ser parametrizado através do uso de generics. Já a estrutura `Triangulo` possui as properties `P1`, `P2` e `P3`, que correspondem aos três pontos que compõem o triângulo. `Triangulo` deve ser parametrizado com o uso de generics, e o tipo parametrizado deve ser utilizado nos pontos do triângulo.

No método `Main()` da aplicação, crie diferentes triângulos e pontos com diversos tipos de dados, a fim de validar a implementação realizada.