# Lista 01 – Primeira Parte – Conceitual

1. O que é Orientação a Objetos?
2. Qual a diferença entre programação Procedural e Orientada a Objetos?
3. O que é uma classe e para que server? Dê exemplos.
4. O que são métodos? Para que servem? Dê exemplos.
5. O que é encapsulamento? Para que Server? Como implementar?
6. O que é um método construtor? Para que server? Como utiliza-lo em um programa?
7. O que são modificadores de acesso? Dê exemplos.
8. O que é Coleção de Objetos em Java?
9. O que é Mapa em Java?
10. Para que server a Classe Object? Como utiliza-la no nosso programa? Dê exemplos.
11. O que é herança? Dê exemplos.
12. O que é Associação? Dê exemplos.
13. Qual a diferença entre Herança e Associação? Dê exemplos.
14. O que é Sobrecarga e Sobrescrita? Dê exemplos.
15. O que é Coesão e Acoplamento? Dê exemplos.
16. O que é Serializable, Transient, Static? Dê exemplos.
17. O que é casting? Dê exemplos.
18. O que é instanceOf? Dê exemplos.
19. O que é Exceções em Java? Dê exemplos.
20. Qual a diferença entre equals, equalsIgnoreCase e compareTo? Dê exemplos.
21. O que é toString? Dê exemplos.
22. O que é hashCode? Dê exemplos.
23. O que é uma classe abstrata? Para que server? Como implementar?
24. O que é uma classe interface? Para que server? Como implementar?
25. O que é arquitetura MVC? Para que server? Como implementar? Dê exemplo.
26. O que é uma classe BO? Para que server? Como implementar? Dê exemplo.
27. O que é uma classe BEAN? Para que server? Como implementar? Dê exemplo.
28. O que é uma classe DAO? Para que server? Como implementar? Dê exemplo.
29. O que é CRUD? Para que server? Como utilizar? Dê exemplos.
30. O que é JDBC?
31. O que é o AWT no Java? Dê exemplos.
32. O que é Swing no Java? Dê exemplos?
33. Qual a diferença entre AWT e Swing? É possível trabalhar em um programa usando AWT e Swing ao mesmo tempo? Dê exemplos.
34. O que é look-and-feel?

# Lista 01 – Segunda Parte – Pratico

1. Faça um programa para ler vários números e informar quantos números entre 100 e 200 foram digitados. Se o valor digitado for 0 o programa deve ser encerrado.
2. Pedro tem 1,50m e cresce 2 centímetros por ano, enquanto Mauro tem 1,10m e cresce 3 centímetros por ano. Construir um algoritmo que calcule e imprima quantos anos serão necessários para que Mauro seja maior que Pedro.
3. Crie um programa em Java que que leia e some valores digitados pelo usuário, a condição de parada é quando o usuário digitar 0. Deverá ser impresso para o usuário a soma dos valores, os valores na ordem que foram digitados, os valores ordenados e a média dos valores.
4. Escreva um programa para mostra a hora atual, colete o nome do usuário logado no pc e uma saudação (bom dia, boa tarde ou boa noite) de acordo com a hora do dia.

**Exemplo**:

12:55 – Boa tarde Luis.

**Dica**: pesquisar pela API System

1. Criei um projeto para calcular dos valores digitados pelo usuário. Deverá ser apresentado um menu para o usuário para que ele possa escolher um dos cálculos abaixo:

* Soma
* Subtração
* Multiplicação
* Divisão
* Resto da Divisão
* Média dos valores

Utilize o conceito de orientação a objetos para implementar.

1. Considere um vetor dinâmico de valores inteiros positivos maiores que zero e um único valor X inteiro positivo maior que zero, construa o seguinte programa:
   1. Carregue um vetor lendo valores do teclado, impossibilitando de o usuário digitar valores negativos a condição de parada é quando o usuário digitar 0.
   2. Leia um valor X digitado pelo usuário e busque no vetor o valor, caso exista o valor no vetor, imprimir o valor e sua posição no vetor. Caso o valor esteja repetido no vetor imprima todas as posições aonde se encontra o valor e informe também a distância entre eles.
   3. Ordenar o vetor, e imprimi-lo em ordem crescente e decrescente.
   4. Informar quantos valores são menores que X, quantos são maiores e quais são iguais.
2. Crie um programa que gere 10 números aleatórios, o método **main()** deve chamar a classe **Aleatorio** para que ela gere tais valores.
3. Desenvolva um projeto que crie uma classe que representa um plano cartesiano, também, crie uma classe para representar um triangulo, reusando a classe anterior por composição. Por fim, escreva um programa que receba do usuário as coordenadas dos vértices do triangulo e imprima seu perímetro.
4. Desenvolva um projeto representar uma conta corrente de um banco. A classe conta corrente deverá ter métodos para depositar, sacar e obter o saldo. Para cada saque será debitada também uma taxa de operação equivalente à 0,5% do valor sacado. Crie, em seguida, uma subclasse desta classe anterior para representar uma conta corrente de um cliente especial. Clientes especiais pagam taxas de operação de apenas 0,1% do valor sacado. Faça testes com as duas classes e verifique seus resultados.
5. Crie uma hierarquia de classes de para uma loja que venda livros, CDs e DVDs. Sobrescreva o método **toString()** para que imprima:

* Para livros: nome, preço e autor.
* Para CDs: nome, preço e número de faixas.
* Para DVDs: nome, preço e duração.

Evite ao máximo repetição de código utilizando a palavra **super** no construtor e no método sobrescrito. Em seguida, crie uma classe Loja com o método **main()** que adicione 5 produtos diferentes (a sua escolha) a um vetor e, por fim, imprima o conteúdo do vetor.

1. Ainda modificando o código do programa anterior, faça com que Produto implemente a interface **Comparable**, e implemente a comparação por nome. Ao final do método **main()**, ordene o vetor utilizando o método **sort()** e imprima-o novamente. Depois altere a implementação da comparação para ordenar por preço e verifique o resultado.
2. Crie a seguinte hierarquia de classes:

* Uma interface para representar qualquer forma geométrica, definindo métodos para cálculo do perímetro e cálculo da área da forma;
* Uma classe abstrata para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados e o método de cálculo do perímetro já pode ser implementado;
* 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;
* Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.

No programa principal, pergunte ao usuário quantas formas ele deseja criar. Em seguida, para cada forma, pergunte se deseja criar um quadrado, um retângulo ou um círculo, solicitando os dados necessários para criar a forma. Todas as formas criadas devem ser armazenadas em um vetor. Finalmente, imprima: (a) os dados (lados ou raio); (b) os perímetros; e (c) as áreas de todas as formas. Para (b) e (c), tire vantagem do polimorfismo, enquanto que para (a) utilize instanceof e downcast.