Resolução – Primeira Lista de Exercícios – P.O.O.

1 - Uma classe é um protótipo que possui variáveis de atribuição (atributos) e funções (métodos). Para se definir uma classe em Java, é necessário:

```
<modificador> class nome_classe <extends superclasse> <implements interface>{
    modificador tipo_var var_atributos;
    modificador tipo_metodos métodos(){}
}
ex: public class Main {
      private float teste; //atributo com modificador private
      public float Teste(){ //método com modificador public
         return teste;
      }
```

Na linguagem C, existe uma característica chamada **struct**. Essa "estrutura" necessita, igualmente nas classes em Java, da criação de variáveis estáticas/dinâmicas. Em Java, as variáveis estáticas são advindas dos parâmetros, e as dinâmicas nas instâncias.

2 - Instância são variáveis do tipo da classe comumente denominadas de objetos. Elas reservam espaço de memória e carregam as rotinas da classe, podendo ser modificadas a seu próprio uso (ex. valor final do atributo do “objA” pode ser diferente do “objB”, mesmo sendo instanciados de uma mesma classe).

Para criá-las, basta:

nome_classe <nome_obj_ou_variável> = new nome_classe();

ex: Main teste = new Main();

No exemplo acima, a linguagem Java reservou espaço de memória do tamanho da classe “Main” para a variável/objeto teste (que irá possuir os atributos e métodos de Main).

3 - Um método sobrecarregado significa que este método possui dois ou mais tipos com mesmo nome, porém com passagem de parâmetros diferentes (exercendo funções diferentes).

ex: public class Main(){

```
    void metodoX(int x){
        x + 2;
    }
    void metodoX(float x){
        x + 3;
    }
}
```

Ou seja, os métodos devem ter o mesmo nome, porém passam parâmetros diferentes.

4 - Override é a capacidade de uma subclasse redefinir um método implementado na superclasse herdada, para se ter o comportamento esperado pelo programador. Em outras palavras, o método da "classe filho" é executado sobre o método com mesmo nome da "classe pai" (superclasse).

Ex: public class A{

```
    void metodoTeste(int x){
        x + 5;
    }
    void metodoTeste2(int y){
        y + 2;
    }
}
public class B extends A{
    void metodoTeste2(float z){
        z + 4;
    }
}
```

No caso acima, ao criar uma instância da classe B e executar o método "metodoTeste2", esse objeto irá executar o método contido na classe B. Contudo, se esse objeto executar o método "metodoTeste", ele irá executar o método contido na classe A (superclasse).

5 - O modificador "final" é a representação de constantes em Java. Ou seja, é utilizado para que o usuário não consiga alterar os atributos ou métodos de uma classe, impedindo o Override (por exemplo).

Ex:
```
public final class Teste {
    final int x;
}
```

6 – Ver as classes abaixo. Utilização dos métodos “padronizados” de Getter e Setter.

```
public class Pessoa {
   private String nome, endereco, telefone;

   public String getNome() {
      return nome;
   }
   public void setNome(String nome) {
      this.nome = nome;
   }

   public String getEndereco() {
      return endereco;
   }
   public void setEndereco(String endereco) {
      this.endereco = endereco;
   }

   public String getTelefone() {
      return telefone;
   }
   public void setTelefone(String telefone) {
      this.telefone = telefone;
   }
```

```
    public void imprimeValores(){
        System.out.println("Nome: " + nome);
        System.out.println("Endereço: " + endereco);
        System.out.println("Telefone: " + telefone);
    }
}

public class Main {
    public static void main(String[] args) {
        //Exercício 6 (sem utilização Scanner)
        Pessoa pessoa = new Pessoa();
        pessoa.setNome("Davi Augusto");
        pessoa.setEndereco("Rua Não Sei Do Que, n° 0 - Vila Uau");
        pessoa.setTelefone("(11) 99999-9999");
        pessoa.imprimeValores();
    }
}
```

7 – Utilização da classe “Pessoa” do exercício anterior para com as classes abaixo.

```
public class Aluno extends Pessoa {
    private int RA;

    public void setRA(int RA){
        this.RA = RA;
    }
    public void imprimeAluno(){
        System.out.println("RA: " + RA);
    }
}
public class Main {
    public static void main(String[] args) {
        //Exercício 7 (sem utilizar Scanner)
        Aluno aluno = new Aluno();
        aluno.setNome("Davi Neves");
        aluno.setEndereco("Rua Não Sei Do Que, n° 0 - Vila Uau");
```

```
        aluno.setTelefone("(11) 00000-0000");
        aluno.setRA(193837099);
        aluno.imprimeValores();
        aluno.imprimeAluno();
    }
}
```

8 - Classes abstratas exigem que seus descendentes herdem os métodos abstratos, ocorrendo modificação destes métodos por cada descendente. Ou seja: cada descendente herda o método e modifica a seu favor.

Ex: Classe Funcionário com o método de bonificação, tendo as Classes Diretor e Secretária com bonificações diferentes (1200$ para o primeiro e 1600$ para a segunda).

```
class Mostra extends MostraDados{

      public void mostra(int l){
          System.out.println(l);
      }
      public void mostra(float f){
          System.out.println(f);
      }
}
```

9 –

```
interface MostraDados {
    void mostra(int l);
    void mostra(float f);
}
class Mostra implements MostraDados{

    public void mostra(int l){
        System.out.println(l);
    }
    public void mostra(float f){
        System.out.println(f);
```

```
    }
}
public class ProgPrincipal {

    //Exercício 8/9
    static int valor = 123;
    static float x = 4.56F;

    static void mostraValores(MostraDados db) {
        db.mostra(x);
        db.mostra(valor);
    }

    public static void main(String[] args) {
        //Exercício 8/9 (Abstract e Interface)
        Mostra mostra = new Mostra();
        mostraValores(mostra);
    }
}
```

10 - A classe está se utilizando de um conceito de POO denominado sobrecarga de métodos, o qual consiste basicamente em criar variações de um mesmo método, porém com argumentos (parâmetros) diferentes para que o sistema possa diferenciar. O Java, durante a compilação, escolhe um dos métodos com base no parâmetro passado pelo objeto.