UNIVERSIDADE PAULISTA – UNIP CAMPUS PARAÍSO

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

ATIVIDADES PRÁTICAS SUPERVISIONADAS

“DESENVOLVIMENTO DE UM PROGRAMA UTILIZANDO

CONCEITOS DE ORIENTAÇÂO A OBJETO”

COMPONENTES DO GRUPO

ANDRE VICTOR DE SOUZA ELIAS D8315B5

JULHO JUSTINO SALES N256940

JULIO CESAR ALCANTARA FREIRE D733CJ0

NAYADNI MARGANELLI D652GH3

VICTOR ADONAI MARASSI DE CARVALHO D668ID9

Classe:

Classe é uma estrutura do Java onde se descreve um objeto, há também a Classe Abstrata onde também se descreve um objeto, mas ela não pode instanciar ou criar um objeto, mas ela é somente usada como herança para outras classes. Dentro da Classe há Atributos e Métodos para instanciar um objeto.

Exemplo:

Public Class Animal {

}

Atributos:

São aqueles que dentro da classe eles descrevem o que o objeto tem de características para ser criado.

Como também podem ser colocado como atributos outras classes como por exemplo quando se cria um sistema de cadastro de pessoas, você pode criar uma classe para a data de nascimento e utilizar como um atributo de outra objeto.

Public Class Animal {

Int Forca=100;

String nome;

}

Métodos:

Métodos também estão dentro da classe, onde eles descrevem o que o classe pode fazer com o objeto, como no método construtor onde o programa cria o objeto.

Métodos contem assinaturas ou não para serem chamados, as assinaturas são atributos que podem ser registrados na criação do objeto.

Como por exemplo o método construtor é o método de instanciação do objeto,

Exemplo:

Public andar () {

Forca=Forca-5;

System.out.printIn(“o animal anda”);

}

Herança:

Classe também tem um sistema de herança onde uma Classe Mae pode passar seus métodos e atributos para a Classe Filha, como numa Classe Mae de pássaro conseguem passar seus métodos e atributos para as Classes Filhas como bem-te-vi e papagaio, como eles tem atributos e métodos diferentes e especializados para cada um, mas tem atributos e métodos iguais a outros pássaros.

Public Class Passaro extends Animal {

Boolean Pena=true;

Public voar () {

If(Pena=true) {

System.out.printIn(“o {0} voa ”+Nome);}

}

Sobrecarga:

Sobrecarga ou (Overload) é um conceito de orientação ao objeto, onde você sobrecarrega um método, sobrecarregar um método é escrever um determinado método e dar para um mesmo método diferentes assinaturas.

Como por exemplo:

Public Class Passaro extends Animal {

Boolean Pena=true;

Public voar () {

If(Pena=true) {

System.out.printIn(“o {0} voa ”+Nome);}

}

Public voar(pena)

If(Pena=true) {

System.out.printIn(“o {0} voa “+Nome);}

}

Sobrescrita:

Sobrescrita ou(Override), é um conceito ligado a herança, pois um método da Classe Mãe pode ser reescrito numa classe filha, diferente de uma sobrecarga a sobrescrita é escrita de maneira idêntica a Classe mãe.

Public Class Animal {

Int Forca=100;

String Nome;

Nascer (String Nome) {

Nome=this.Nome;}

}

Public Class Passaro extends Animal {

Boolean Pena=true;

Nascer (String Nome, Int Forca) {

Nome=this. Nome;

Forca=this.Forca;

Public voar () {

If (Pena=true) {

System.out.printIn (“o {0} voa ”+Nome);}

}

Encapsulamento:

Vem de encapsular que significa isolar um determinado objeto, em programação orientada a objeto ele divide o programa em partes.

Dentro do encapsulamento temos 4 métodos de encapsular o programa:

1. Private é o método de encapsulamento que impede o programa de ler e alterar o programa livremente, com isso os únicos métodos capazes de alterar o estado do atributos são os métodos gets e sets. Este método de encapsulamento não pode ser usado em uma classe, apenas em seus métodos e atributos.
2. Protected é o método de acessibilidade de um programa em Java, onde ele permite que a classe, atributos e seus métodos sejam herdados por todas as classes do pacote do Java Project.
3. Public é o método de acesso livre para todos as classe que possam visualizar a classe.
4. O método padrão(default) é definido quando não é escrito nada na classe sendo um método padrão do compilador.

Polimorfismo:

Conhecido também como terceiro pilar da programação orientado a objeto e também complementado pelos conceitos de herança e sobrecarga.  
 O Polimorfismo vem de uma palavra grega, onde “poli” significa muitos, vários, e “morfismo” esta relacionando à forma, tornando polimorfismo significar (“muitas formas”), em programação orientada ao objeto ela permanece com esse significado, dando a possibilidade de reutilizar os mesmos métodos em várias partes do programa.   
 Um exemplo dessa reutilização seria se a gente criasse um método de “abrir ()”, podemos usar esse método para abrir uma garrafa, uma porta, uma janela, entre outras. Outro exemplo seria usando um método “comunicar ()” onde usaríamos criança, gato, pássaro, gato como objeto, no qual poderíamos usar o método comunicar em todos, claro que teria que ter algumas alterações por eles terem diferentes formas de comunicação, porem o método estaria presente em todos.   
 A grande vantagem de usar Polimorfismo é que se precisar de alguma manutenção ou atualização é só ir direto para o código raiz e resolver o problema.

Interface:

Interface, é um mecanismo simplificado que possibilita a implementação de heranças múltiplas em Java, também conhecida por ser bem semelhante à classe abstrata, diferenciando que todos os métodos serão abstratos.  
 Assim como na classe abstrata, uma interface não pode ser instanciada, e todos seus métodos tem que ser public ou abstract.  
 A principal diferença entre elas é que uma classe herdeira só pode ser usada uma vez, e uma classe com interface pode ser usada inúmeras vezes durante o programa.  
 Interfaces só pode ser utilizada pelo níveis de acesso public e default.