**RESUMO DA AULA 05 - HERANÇA**

ALUNO: Denilson José do Bom Jesus Silva de Lima

- Herança

Conceito que trabalha com a ideia de herdar o código da Classe "pai" para a classe "filha", evitando a necessidade de copiar códigos e mesmo assim mantendo

uma integração entre os códigos, ou seja, permite reutilizar códigos de classes já existentes adicionando assim somente o que é necessário para a nova classe.

É um relacionamento e hierarquia entre classes, mas não entre objetos distintos.

Superclasse e Subclasse: tudo que a Superclasse tem a Subclasse também terá. Segundo o princípio da Extensibilidade algumas operações da superclasse podem ser redefinidas no subclasse. No Comportamento os objetos da subclasse se comportam como objetos da superclasse e de acordo com o Princípio da Substituição os objetos da subclasse podem ser usados no lugar de objetos da superclasse.

A única coisa não herdada na classe filha são os construtores que precisam ser implementadas; os construtores da subclasse sempre utilizam algum construtor

da superclasse.

Sintaxe (Ex:)

public class NomeClasseFilha extends NomeClassePai {

...

}

A Classe Filha tem os mesmos atributos Private da Classe Pai, porém ela não tem acesso a eles diretamente através do ".this". Ela pode usar esses atributos com métodos usando "get" e "set" ou chamando-os com "super()".

Quando não se declara de qual Classe uma classe descende, de acordo com o default, por padrão, se descende da classe "Object" - que é a única classe de Java

que não herda nenhuma outra classe; é a superclasse de todas as classes de Java. (Ex:)

public class NomeClasse extends Object {

...

}

- Sobre o Construtor

Se não se estabelecer o construtor (ou se tiver um construtor sem parâmetros com o "super()" na classe Pai), é estabelecido o construtor vazio - com uma linha de código que chama o construtor vazio da classe Pai, super(), na classe Filha. Ao se definir o construtor, não se tem mais o construtor vazio.

Caso o construtor esteja estabelecido na classe Pai é preciso fazer a chamada na classe Filha - a chamada ao "super()" no construtor aparece uma vez na primeira linha, já nas chamadas a métodos da classe Pai, é possível fazer o super() várias vezes, de acordo com a necessidade.

- Cast

Conversão dinâmica de tipos, nos tipos primitivos pode ser implícita (promoção aritmética) ou explícita (perda de precisão), já nos tipos referenciados segue a ideia de "do tipo mais geral para o tipo mais específico o cast deve ser explícito." (Ex:)

ClassePai classe1 - new ClasseFilho(); - em tempo de execução é associado como uma ClasseFilho, mas o compilador interpreta como sendo da ClassePai (visto que Java é uma linguagem tipada - sendo para o compilador a ordem da esquerda para a direita importante).

Para fazer o compilador entender que deve associar o novo objeto a ClasseFilho e não a ClassePai, utiliza-se o Cast (Ex:)

((ClasseFilho)classe1).setExemplo(); - normalmente se usa esse tipo de comportamento quando se herda de biblioteca de terceiros.

Um Cast compila quando o tipo da variável de origem e o tipo do cast estão na mesma hierarquia e quando o tipo de cast é igual ou sub-tipo da variável de destino. E roda quando compila e quando o tipo do objeto para o qual a variável de origem aponta é igual ou sub-tipo do tipo de cast.

- Instanceof

Operador que verifica a classe de um objeto (retorna true ou false), recomenda-se o uso antes de um cast para evitar erros em tempo de execução.