Aula 5

Herança em Java

Em Java, subclasses são declaradas através do uso da palavra-chave `extends`, como no exemplo a seguir:

```
class Janela {
   // ...
};

class JanelaTexto extends Janela {
   // ...
};
```

Subclasses são imaginadas como sendo subtipos (embora isto não seja sempre válido). Isto significa que uma instância de uma subclasse pode ser associada a uma variável declarada com o tipo da classe pai. Métodos na classe filha que possuem o mesmo nome daqueles na classe pai sobrescrevem o comportamento herdado. Como em C++, a palavra-chave `protected` pode ser usada para designar métodos e dados que são acessíveis apenas dentro da classe ou dentro de subclasses, mas não fazem parte da interface mais geral.

Todas as classes são derivadas de uma única classe raiz: `Object`. Se nenhum pai é explicitamente fornecido, a classe `Object` é assumida. Portanto, a declaração de classe para `Janela` mostrada acima é a mesma que a seguinte:

```
class Janela extends Object {
   // ...
};
```

Uma forma alternativa de gerar subtipos envolve o uso de *interface*. Uma interface define o protocolo para certo comportamento, mas não a implementação. Neste sentido, interfaces são semelhantes a classes pais abstratas. Segue um exemplo de interface, descrevendo objetos que podem ler e escrever para um buffer de entrada e saída:

```
public interface Armazenamento {
   void escreve(Stream s);
   void le(Stream s);
};
```

Uma interface define um novo tipo. Isto significa que variáveis podem ser declaradas simplesmente pelo nome da interface. Uma classe pode então indicar que ela implementa o protocolo definido por uma interface. Instâncias da classe podem ser associadas a variáveis declaradas com o tipo da interface, assim como instâncias de uma classe filha podem ser associadas a variáveis declaradas com o tipo da classe pai.

```
public class ImagemBinaria implements Armazenamento {
   void escreve(Stream s)
   {
       \\ ...
   }

   void le(Stream s)
   {
       \\ ...
   }
}
```

Embora Java suporte apenas herança simples (herança a partir de uma única classe pai), uma classe pode indicar que ela implementa múltiplas interfaces. Muitos problemas para os quais herança múltipla poderia ser usada em C++ podem ser resolvidos em Java pelo uso de múltiplas interfaces, pois interfaces podem "estender" outras interfaces.

A idéia de herança por especialização é formalizada em Java através do modificador `abstract`. Se uma classe for declarada com `abstract`, deve-se criar uma filha desta classe, pois não é permitido criar uma instância de uma classe abstrata, apenas subclasses. Métodos também podem ser declarados como `abstract` e, se isso acontecer, eles não necessitam ser implementados. Portanto, definir um método como `abstract`

faz com que ele possa ser usado apenas como uma especificação de comportamento, não como uma implementação.

```
abstract class persistente {
   public abstract void escreve();

}
```

Um modificador alternativo, `final`, indica que uma classe (ou método) não pode ser herdada ou modificada. Portanto, o usuário está seguro que o comportamento da classe será como definido e não será modificado por uma subclasse que venha a ser criada depois.

```
final class NovaClasse extends VelhaClasse {
   ...
}
```