Gabarito das Autoatividades

PROGRAMAÇÃO ORIENTADA A OBJETOS

UNIASSELVI

Centro Universitário Leonardo da Vinci
Rodovia BR 470, Km 71, nº 1.040
Bairro Benedito - CEP 89130-000
Indaial - Santa Catarina - 47 3281-9000

Elaboração:
Prof. Cristiano Roberto Franco

Revisão, Diagramação e Produção:
Centro Universitário Leonardo da Vinci - UNIASSELVI

GABARITO DAS AUTOATIVIDADES DE PROGRAMAÇÃO ORIENTADA A OBJETOS

UNIDADE 1

TÓPICO 1

1 Assinale a alternativa CORRETA:

a) (x) Uma classe é uma espécie de forma que vai definir tributos e/ou comportamentos de todos aqueles objetos que forem instanciados a partir dela.
b) ( ) A POO (programação orientada a objetos) é equivalente à programação estruturada, pois se fundamenta no relacionamento entre variáveis e classes, através de polimorfismo e interfaces.
c) ( ) Um objeto é a representação prática de um método de uma classe, pois todo método precisa retornar um tipo de informação.
d) ( ) A partir do momento em que criamos um objeto, temos um novo TIPO disponível para utilização.

2 Descreva três vantagens da programação orientada a objetos sobre a programação Estruturada.

R.: A programação orientada a objetos é mais natural e relaciona-se mais facilmente ao mundo real, o que facilita sua compreensão. Ela permite a reutilização através da herança e facilita a manutenção futura através do polimorfismo.

3 Assinale a alternativa CORRETA:

a) ( ) As mensagens trocadas entre os objetos são conhecidas como atributos.
b) ( ) Os atributos e os métodos dos objetos são conhecidos como o ESTADO dos mesmos.
c) ( ) A programação orientada a objetos exige a utilização de uma máquina virtual que proteja o sistema operacional de possíveis erros de um programa.
d) (x) O encapsulamento consiste em proteger os dados de um objeto do acesso externo, liberando-os conforme houver necessidade.

4 Observe as afirmações a seguir, classificando-as em verdadeiras (V) ou falsas (F):

(F) A linguagem de programação Java teve sua arquitetura baseada na arquitetura da linguagem de programação C++.
(V) Para uma linguagem ser considerada Orientada a objetos, ela deve implementar os conceitos de encapsulamento, herança e polimorfismo.
(V) A extensibilidade dos objetos facilita seu reúso por outros programadores.
(V) A plataforma Java é composta por uma máquina virtual, uma linguagem de programação, um conjunto de bibliotecas e um mecanismo de instalação ou *deployment*.
(V) O JRE é o ambiente de execução das aplicações Java, enquanto o JDK reúne um conjunto de ferramentas para desenvolvedores.
(V) O Java é considerado uma linguagem de programação compilada e interpretada. Ela é compilada pelo compilador javac e depois tem seus *bytecodes* interpretados pela JVM.
(V) *Hotspot* é o nome do recurso da JVM para otimizar *bytecode* em tempo de interpretação.

5 Descreva as principais diferenças entre um programa compilado e um interpretado.

R.: Em um programa compilado, o compilador checa todo o código antes de gerar um binário para determinada plataforma, enquanto em um programa interpretado, a verificação de erros é feita ao mesmo tempo em que as linhas são interpretadas sequencialmente.

TÓPICO 2

1 De acordo com os procedimentos vistos neste tópico, crie um projeto no Eclipse chamado de Exercício, contendo um pacote chamado topico 1:

R.: Resposta do acadêmico, individual.

2 Dentro do projeto criado na questão 1, crie mais dois pacotes, chamados topico 2 e topico 3.

R.:Resposta do acadêmico, individual.

3 Dentro dos pacotes topico 1 e topico 3 crie uma classe chamada de Classe Teste, e dentro do pacote topico 2, crie um subpacote chamado de topico 2.1. Dentro de topico 2.1 crie outra classe chamada Classe Teste.

R.: Resposta do acadêmico, individual.

4 Nas linhas a seguir escreva o *fully qualified name* de todas as classes existentes no projeto Exercício.

R.: topico1.Teste
topico3.Teste
topico2.topico2.1.Teste

5 Encontre o diretório do projeto dentro de seu *workspace* e liste todos os subdiretórios do projeto nas linhas a seguir:

R.:Resposta do acadêmico, individual.

6 Qual é o objetivo da existência dos diretórios bin e src?

R.: O diretório bin abriga os binários das classes, representados pelos arquivos .class, enquanto o diretório src abriga os códigos-fonte das classes, representados pelos arquivos .java

---

TÓPICO 3

1 Avalie as afirmações a seguir, colocando V para as afirmações Verdadeiras e F para as afirmações Falsas:

( ) Os atributos de uma classe definem o comportamento dela.
( ) Todos os métodos devem retornar um tipo de dado após terminarem sua execução.
( ) Os símbolos { e } representam o início e o fim de um bloco de código-fonte, que pode ser uma classe, um método etc.
( ) A declaração de uma variável ou um atributo utilizando como tipo uma classe recém-criada somente pode acontecer após a instanciação.
( ) Gerar Relatório é um nome adequado para uma classe, uma vez que é significativo em relação ao significado.

Agora assinale a alternativa que contém a sequência CORRETA:
a) F, F, V, F, F.
b) F, V, V, F, F.
c) V, F, V, F, F.
d) F, F, V, F, V.

2 Utilizando os procedimentos mostrados neste tópico, crie no Eclipse um projeto com o nome Veterinário que contenha um pacote chamado canil e uma classe chamada Cachorro, contendo as seguintes informações:

• Atributos: nome, raça e peso.
• Métodos para coçar (sem implementação) e para latir, imprimindo na saída padrão a mensagem (“au...au...au”).

R.: Resposta do acadêmico, individual.

3 Utilizando o projeto do exercício anterior, crie dentro do pacote canil uma classe chamada de Testadora, através da qual você deve:

• Instanciar um objeto do tipo Cachorro.
• Atribuir valores para os atributos do Objeto.
• Invocar o método latir() do objeto instanciado e verificar o resultado na saída padrão.

R.: Resposta do acadêmico, individual.

4 Utilizando o projeto do exercício anterior, crie um novo pacote chamado de curral e, dentro deste pacote, crie uma classe que represente um tipo de animal. Esta classe deve possuir atributos e métodos condizentes com o animal que ela representa.

R.: Resposta do acadêmico, individual.

UNIDADE 2

TÓPICO 1

1 Assinale a alternativa CORRETA:

a) O encapsulamento não pode ser definido em nível de classe, pois não pode existir uma classe private.
b) Existem três modificadores de visibilidade no Java:
• public libera o acesso do item em questão (classe, método, atributo) a TODAS as demais classes da aplicação;
• implemented libera o acesso do item em questão (classe, método, atributo) somente a classes que forem derivadas (filhas) da classe onde o modificador foi aplicado;
• private libera o acesso do item em questão (método, atributo) somente a classe onde ele está inserido.
c) Um mesmo arquivo-fonte escrito em Java não pode ser compilado em diferentes sistemas operacionais, cada um com sua máquina virtual específica.
d) Encapsulamento consiste em ocultar parte de seus dados do resto de sua aplicação e limitar a possibilidade de outras partes de seu código acessarem esses dados. Ao invés de ter um programa como uma entidade grande e monolítica, o encapsulamento permite que você o divida em várias partes menores e independentes, facilitando manutenções futuras.

2 Descreva de que forma a alta coesão e o baixo acoplamento contribuem para que seja mais fácil fazer a manutenção de programas de computador.

R.: A alta coesão permite que cada classe tenha uma responsabilidade única, o que facilita o processo de localização do erro, caso algum problema ocorra. Já o baixo acoplamento garante que, caso uma modificação em determinada parte do sistema seja necessária, as demais partes não sejam penalizadas com alterações em cascata.

3 Assinale a alternativa CORRETA:

a) O encapsulamento é implementado através de modificadores de visibilidade. Ao identificar uma classe, método ou campo com eles, define-se o que será oculto e o que será visível às demais classes da aplicação.

b) Uma classe pode conter somente um método construtor.
c) A linguagem de programação Java é considerada multiplataforma pelo fato de podermos escrever programas Java tanto em um editor de texto comum quanto em IDES mais elaboradas, como, por exemplo, Eclipse ou Netbeans.
d) Um mesmo arquivo .class Java que foi compilado e transformado em *bytecode* no Linux pode ser simplesmente copiado e executado em qualquer outro sistema operacional, mesmo sem máquina virtual (JVM).

4 Observe as afirmações a seguir, classificando-as em verdadeiras (V) ou falsas (F):

(F) Uma classe somente pode ser constituída por um nome e por um conjunto de métodos que definem o seu comportamento.
(F) Uma classe não pode conter mais de um método construtor, pois no momento da instanciação o compilador não saberia qual construtor chamar.
(V) Uma classe pode possuir uma variedade especial de métodos que são chamados de construtores. A principal funcionalidade desses métodos é construir a classe, atribuindo valores aos campos etc.
(F) Encapsulamento consiste em colocar TODOS os seus atributos e métodos como *private*, impedindo o acesso a estes por outras partes de sua aplicação. Ao invés de ter um programa como uma entidade grande e monolítica, o encapsulamento permite que você o divida em várias partes menores e independentes, facilitando manutenções futuras.
(F) Uma variável local em Java sempre deve ser inicializada e pode conter qualquer modificador (*private, public, protected*), pois seu escopo se alterará de acordo com o modificador aplicado.

5 Explique detalhadamente porque não faz sentido colocar o modificador *public* em um método dentro de uma classe com visibilidade *default*.

R.: Não faz sentido porque a classe nunca seria vista fora de seu pacote, o que impediria o modificador public de exercer seu efeito de disponibilizar o método para toda a aplicação.

TÓPICO 2

1 Crie uma hierarquia de herança fictícia, desenhando o diagrama da UML e descreva textualmente quais atributos e métodos da superclasse serão herdados pelas subclasses.

R.:Resposta do acadêmico, individual.

2 Crie um projeto no Eclipse chamado de HERANCA e implemente todas as classes que foram definidas no exercício 1, incluindo seus atributos e métodos. Crie ainda uma classe testadora para testar a criação dos objetos e seus relacionamentos.

R.:Resposta do acadêmico, individual.

3 Com relação ao relacionamento de herança, assinale V para as alternativas VERDADEIRAS e F para as alternativas FALSAS:

(F) O relacionamento de herança é semelhante ao relacionamento de associação.
(F) A superclasse ou classe filha é a classe que fornece o estado ou comportamento a ser herdado pelas subclasses.
(F) Para se definir o relacionamento de herança, basta fazer a pergunta: "é um tipo de?"
(V) Uma subclasse pode estar relacionada a uma e somente uma superclasse.
(V) A Herança é uma prática que pode interferir com o encapsulamento entre as classes.

4 Altere o diagrama feito no exercício 1 de forma a eliminar a herança e incluir a composição no relacionamento entre as classes.

R.:Resposta do acadêmico, individual.

5 Crie um projeto no Eclipse chamado de COMPOSICAO e implemente todas as classes que foram definidas no exercício 4, incluindo seus atributos e métodos. Crie ainda uma classe testadora para testar a criação dos objetos e seus relacionamentos.

R.:Resposta do acadêmico, individual.

6 Descreva detalhadamente de que forma a composição elimina parte dos problemas causados pela herança.

R.: A composição permite a reutilização de código-fonte sem quebrar o encapsulamento da classe mãe, além de permitir que se "herde" atributos e métodos de inúmeras classes. Outro ponto positivo é a questão do princípio de substituição de Liskov, que não se aplica a relacionamentos de composição, eliminando parte da complexidade da herança. Finalmente, um relacionamento de composição torna mais fácil realizar mudanças na parte composta e na parte que está compondo, pois é menos invasivo que um relacionamento de herança.

TÓPICO 3

1 Avalie as afirmações a seguir, colocando V para as afirmações Verdadeiras e F para as Falsas:

( ) Polimorfismo significa colocar diversos comportamentos sob um mesmo nome, escolhidos em tempo de compilação.
( ) Para que o polimorfismo aconteça, existe a obrigatoriedade da herança.
( ) Uma classe abstrata é uma classe que obrigatoriamente possui métodos abstratos.
( ) Uma interface não pode conter métodos concretos.
( ) Quando uma classe implementa uma interface, a partir deste momento podemos tipá-la através desta interface.

Agora assinale a alternativa que contém a sequência CORRETA:
a) (x) F – F – F – V – V.
b) ( ) F – V – V – F – F.
c) ( ) V – F – V – F – F.
d) ( ) F – F – V – F – V.

2 Utilizando as classes criadas no exemplo da Figura Geométrica visto neste tópico, refatore o código de modo a remover o método abstrato calcularArea() da classe Figura Geometrica e o coloque dentro de uma interface. Deve ser criada uma classe testadora que permita o teste de todas as figuras.

R.:

Interface IGeométrica com o método abstrato

```
1
2 public interface IGeometrica {
3
4     double calcularArea();
5
6 }
```

Classe FiguraGeometrica implementando a nova interface

```
2 public abstract class FiguraGeometrica implements IGeometrica{
3
4 }
```

Classe Quadrado sobrepondo o método calcularArea vindo da interface

```
1
2 public final class Quadrado extends FiguraGeometrica {
3
4     @Override
5     public double calcularArea() {
6
7         return 0;
8     }
9
10 }
```

Teste do método

```
1
2 public class Testador {
3
4     public static void main(String[] args) {
5
6         IGeometrica quadrado = new Quadrado();
7
8         double area = quadrado.calcularArea();
9
10
11    }
12
13 }
```

3 Qual é a principal utilidade das interfaces quando em comparação com as classes abstratas? Dê exemplos:

R.: As interfaces são mais flexíveis quando comparadas com as classes abstratas, pois o relacionamento é menos invasivo. Além disso, uma classe pode implementar quantas interfaces desejar e herdar de somente uma classe abstrata. Uma interface define o que uma classe faz, ao invés do que

uma classe é. Finalmente, uma interface obriga a implementação de um comportamento, sendo comparada à assinatura de um contrato.

4 Crie um novo projeto no Eclipse e desenvolva um exemplo contendo classes para demonstrar de que forma o polimorfismo pode auxiliar na manutenção do código-fonte de uma aplicação. O exemplo deve ser diferente daqueles mostrados nesta unidade.

R.:Resposta do acadêmico, individual.

UNIDADE 3

TÓPICO 1

1 Assinale a alternativa CORRETA:

a) Um Hashset considera a ordem de inserção, quando se procura buscar determinado elemento.
b) Existem três maneiras de se ordenar coleções no Java:
• Implementação da interface Comparator na classe que se deseja ordenar.
• Implementação da interface Comparable em um provider.
• Através de lambda expressions.
c) O método equals determina a igualdade entre dois objetos com base em seu endereço na memória.
d) O método hashcode é utilizado para auxiliar o espalhamento dos objetos nas tabelas hash.

2 Descreva linha por linha a funcionalidade do método equals mostrado no Quadro 69 deste tópico.

R.:

```
35      @Override
36      public boolean equals(Object obj) {
37          if (this == obj)
38              return true;
39          if (obj == null)
40              return false;
41          if (getClass() != obj.getClass())
42              return false;
43          Pessoa other = (Pessoa) obj;
44          if (cpf == null) {
45              if (other.cpf != null)
46                  return false;
47          } else if (!cpf.equals(other.cpf))
48              return false;
49          return true;
50      }
```

Na linha 37 o método pergunta se a instância atual é a mesma do objeto que veio como parâmetro, neste caso os dois são iguais. Na linha 39 o método pergunta se o parâmetro é nulo, neste caso os dois são diferentes. Na linha 41 é feita a comparação do tipo de classe de cada um. Na linha 43 é feita uma conversão de object para Pessoa. Na linha 44 até o final, verifica-se se o atributo CPF do objeto atual é igual ao atributo CPF do objeto que veio como parâmetro. Se os atributos forem iguais, retorna-se true, caso contrário, false.

3 Explique detalhadamente as diferenças entre as três coleções a seguir:

a) Hashset. Os objetos são armazenados em uma espécie de conjunto, onde não há ordenação e não são aceitas duplicatas.
b) Hashmap. Os objetos são armazenados em elementos do tipo chave/valor, onde tanto a chave quanto o valor podem ser do tipo object. Não é ordenado e não aceita duplicatas de valor de chave.
c) ArrayList. Os objetos são armazenados em uma espécie de vetor dinâmico, que pode ser ordenado e aceita duplicatas.

4 Crie um exemplo de código-fonte onde uma coleção de objetos do tipo automóvel é ordenada de forma crescente pela placa através de um provider. Os demais atributos são marca, quilometragem e modelo. Crie um testador e as demais classes de forma a demonstrar a ordenação e criação do critério para tal.

```java
public class Automovel {
    private String marca;
    private String modelo;
    private long kilometragem;
    private String placa;

    public Automovel(String marca, String modelo, long kilometragem,
            String placa) {
        this.marca = marca;
        this.modelo = modelo;
        this.kilometragem = kilometragem;
        this.placa = placa;
    }

    public String getPlaca() {
        return placa;
    }

    @Override
    public String toString() {
        return "Automovel [marca=" + marca + ", modelo=" + modelo
                + ", kilometragem=" + kilometragem + ", placa=" + placa + "]";
    }
}
```

```java
public class ProviderPlaca implements Comparator<Automovel>{

    @Override
    public int compare(Automovel arg0, Automovel arg1) {
        return arg0.getPlaca().compareTo(arg1.getPlaca());
    }

}
```

```java
public class TestadorAutomovel {
    public static void main(String[] args) {
        Automovel auto1 = new Automovel("Porsche", "Carrera", 1234, "crf-4321");

        Automovel auto2 = new Automovel("VolksWagen", "Brasilia", 178987, "eew-8765");

        Automovel auto3 = new Automovel("Fiat", "147", 234590,"qki-4999");

        List<Automovel> lista = new ArrayList<>();

        lista.add(auto3);
        lista.add(auto1);
        lista.add(auto2);

        Collections.sort(lista, new ProviderPlaca());

        System.out.println(lista);

    }
}
```

TÓPICO 2

1 Um dos problemas do padrão de projetos singleton ocorre porque, em caso de sistemas multithread, se duas threads acessarem o método getInstance simultaneamente, pode ser que mais de uma instância seja criada. Pesquise sobre o tema, implemente um singleton que seja seguro em relação a threads simultâneas no Eclipse e justifique sua solução nas linhas a seguir.

```
3  public class Singleton {
4
5      private static Singleton instancia;
6
7      private Singleton() {}
8
9      public static synchronized Singleton getInstance(){
10         if (instancia == null)
11             instancia = new Singleton();
12         return instancia;
13     }
14 }
```

A palavra synchronized não permite que a thread que esteja executando o método seja interrompida até que o mesmo encerre. Desta forma temos a garantia de que somente existirá uma instância mesmo em ambiente multitarefa.

2 No exemplo de exceção personalizada, utilizamos uma característica da linguagem de programação Java que ainda não havia sido mostrada: as enumerações. Pesquise sobre o tema e implemente no Eclipse uma classe que faça uso de uma enumeração chamada de Sexo, que contenha os valores MASCULINO E FEMININO.

```
3 public enum Sexo {
4
5         MASCULINO,
6         FEMININO;
7 }
```

```
3 public class Pessoa {
4
5
6     private String nome;
7     private Sexo sexo;
8
9     public Pessoa(String nome, Sexo sexo) {
10        super();
11        this.nome = nome;
12        this.sexo = sexo;
13    }
14
15    public String getNome() {
16        return nome;
17    }
18
19    public Sexo getSexo() {
20        return sexo;
21    }
22
23
24 }
```

3 Com relação ao conteúdo do Tópico 2, assinale V para as alternativas VERDADEIRAS e F para as FALSAS:

(F) O modificador static pode ser utilizado quando precisamos de uma variável global.
(V) O modificador static pode ser aplicado em métodos e atributos.
(V) Métodos estáticos são normalmente característicos de classes utilitárias que não precisam de estado.
(F) Uma exceção somente é disparada quando ocorre um erro no código-fonte.
(F) Uma ArithmeticException é um tipo de exceção verificada.

4 Descreva detalhadamente por que a utilização de métodos e atributos do tipo static pode, em algumas situações, ser nociva para seu código-fonte.

R.: Atributos e métodos estáticos, apesar de úteis em determinadas circunstâncias, devem ser utilizados com parcimônia. Atributos estáticos são considerados como variáveis globais, e devem ser evitados a todo custo, pois representam uma violação do encapsulamento. Além disso, atributos estáticos fazem com que a classe onde estejam inseridos leve mais tempo para ser coletada pelo garbagecollector, aumentando o consumo de memória de sua aplicação.

5 O Singleton é um padrão de projeto desenvolvido para uma finalidade específica. Pesquise sobre o tema Padrões de Projeto e implemente um deles no Eclipse, justificando a sua aplicabilidade.

R.: O padrão de projetos Proxy serve para isolar uma implementação da qual não se tem acesso ao código-fonte ou mesmo não se quer mexer, fornecendo um novo resultado ou retorno para um método.

```
3  public class CalculoImexivel {
4
5      public double calcular(double valor){
6          valor = valor * Math.ceil(Math.random()+13);
7          return valor;
8
9      }
10
11
12 }
```

```
3  public class ProxyCalculoImexivel {
4
5      private CalculoImexivel ci;
6
7      public ProxyCalculoImexivel (CalculoImexivel ci){
8          this.ci=ci;
9      }
10
11     public double calcular(double v){
12         double valor = ci.calcular(v);
13
14
15         valor = valor*0.12; // encapsula a operacao com seu calculo particular
16         return valor;
17     }
18 }
```

TÓPICO 3

1 Em nosso exemplo de inserção de objetos no banco de dados utilizamos o construtor vazio e setters para atribuir valores para o livro. Por que não utilizamos o construtor completo do objeto?

R.: Por que não sabemos o valor do id do objeto no momento de sua criação, pois este é gerado pelo banco de dados. No caso de recuperação de objetos do banco, poderíamos utilizar o construtor padrão, afinal já temos o valor.

2 Sempre que fazemos uso de um recurso do sistema operacional através de uma conexão, devemos fechar esta conexão assim que terminarmos nossas tarefas. No Quadro 101 fazemos esse fechamento através de um método específico, que inclusive não está implementado na classe LivroDAO. Implemente esse método e altere o exemplo do Quadro 92 para que façamos o fechamento da conexão após inserir os dois livros no banco de dados.

R.: Inicialmente criamos um método para obter a conexão no LivroDAO.

```
    public Conexao getConexao() {
        return conexao;
    }
```

Depois simplesmente fechamos a conexão após encerrar as operações (linha 22).

```
5  public class Testador {
6
7      public static void main(String[] args) {
8          LivroDAO dao = new LivroDAO();
9
10         System.out.println(dao.list());
11
12         Livro l = dao.getLivro(25);
13         l.setPaginas(456);
14         dao.update(l);
15
16         System.out.println(dao.list());
17
18         dao.remove(l);
19
20         System.out.println(dao.list());
21
22         dao.getConexao().fecharConexao();
23     }
24 }
```

3 Na Figura 72 mostramos a impressão dos objetos que retornaram com o método list() da classe LivroDAO. Se você fizer os mesmos testes, perceberá que seu resultado no console é ligeiramente diferente. Descubra o motivo e faça a modificação necessária para que seu exemplo funcione exatamente ao do caderno.

R.: Basta criar o método toString na classe Livro.

```
86
87     @Override
88     public String toString() {
89         return "Livro [titulo=" + titulo + "]";
90     }
91
92
```