Universidade da Beira Interior

Programação Orientada a Objectos
Cursos: 1º ciclo em Eng.ª Informática
Frequência, 2022/12/12

Sem consulta. Não é permitido durante o teste ter auricular, smartwatch ou telemóvel.
Duração: 120 minutos, 13.0 valores

1 – Considere a classe Viagem cujos atributos estão representados abaixo:

```
public class Viagem {
       private final String [] trs = {"Avião", "Autocarro", "Barco", "Comboio" };
       private static int ultimoId=0;
       private int id;
       private String origem;
       private String destino;
       private String transporte;
       private double custo;
```

…

Uma viagem tem um identificador (**id**) gerado automaticamente de forma sequencial. A variável **ultimoId** representa o último identificador atribuído a uma viagem. Uma viagem tem uma **origem**, um **destino**, um meio de **transporte** e um custo. A constante **trs** representa alguns meios de transporte.

a) Para a classe Viagem defina um construtor que receba como parâmetro a origem e o destino da viagem. O construtor deve atribuir o valor "Comboio", à variável **transporte** e gerar o id da viagem de forma sequencial.

b) Construa o método toSring que deve sobrepor o método toString da casse Object (public String toString()).

c) Construa os getters e setters para as variáveis **ultimoID** e **iD.**
Suponha que os outros getters e setters estão já construídos.

d) Supondo que consideramos que dois objetos do tipo viagem são iguais se tiverem a mesma origem o mesmo destino, o mesmo transporte e o mesmo custo, construa o método equals para a classe Viagem. Deve sobrepor o método equals da classe Object (public boolean equals (Object).

e) Suponha uma classe de teste com o programa abaixo:

```
public static void main (String args[]) {
        Viagem v1 = new Viagem ("Lisboa", "Porto" );
        v1.setCusto( 30);
        System.out .println (v1);

        Viagem v2 = new Viagem ("Lisboa", "Faro");
        v2.setCusto( 30);
        System.out .println (v2);

        Viagem v3 = v2;
        v3.setDestino("Porto");

        System.out .println (v3);
        System.out.println (v1 == v2);
        System.out.println (v1.equals(v2));
        System.out.println (v2 == v3);
        System.out.println (v2.equals(v3));
```

i) Quais as variáveis que existem no programa?

ii) Qual o seu output?

f) Suponha agora que queremos alterar o método setTransporte para que verifique se o transporte dado como parâmetro é um transporte válido, isto é, se é um dos valores do array **trs**. Caso não seja um transporte válido, o método deve gerar a exceção TransporteException com a mensagem de erro `"Esse " + XXXXX + " não existe"`, onde xxxxx representa o valor recebido como parâmetro do método. A exceção será tratada no programa que invocar a função.

g) Construa a classe de exceção TransporteException.

2 – Para a classe de teste construa um método (public static) que receba como parâmetro uma lista de viagens, (ArrayList<Viagem>) e que devolva o custo total de todas as viagens da lista.

3 – Para a classe de teste construa um método (public static) que receba como parâmetros uma lista de viagens, uma origem e um destino. O método deve determinar e devolver qual o transporte da viagem mais barata entre a origem e o destino dados.
Notas: inicialize o valor do custo mínimo da viagem com Integer.MAX_VALUE. *Se existirem na lista várias viagens com o custo mínimo, o método apenas devolve o transporte de uma delas.*

4 – No programa do exercício 1:

i) Teste o método setTransporte alterando o meio de transporte da viagem **v1** para um valor dado pelo utilizado. Para ler valores do teclado utilize a classe Ler;
ii) Teste o método do exercício 2;
iii) Teste o método do exercício 3.

5 – Considere que foi implementada a classe Pessoa esquematizada abaixo. Além de um construtor, dos getters e setters, do toString e do clone possui o método **idade** que devolve a idade da pessoa.

```
public class Pessoa {
  private String nome;
  private LocalDate dataNascimento;
  private int NIF;

  public Pessoa (String nome, LocalDate dataNascimento) { …  }
  public String getNome() { … }
  public void setNome(String nome) { …}
  public LocalDate getDataNascimento() { …}
  public void setDataNascimento(LocalDate dataNascimento) {…}
  public int getNIF() {…}
  public void setNIF(int nIF) { …}
  public String toString() { …}
  public Object clone() { …}

  public int idade () {…}
```

a) Queremos agora implementar uma classe que represente uma viagem de grupo ou excursão. Uma excursão é uma viagem organizada para um conjunto de pessoas, podendo ou não incluir alojamento. Defina o cabeçalho e os atributos da classe Excursao tendo em conta que uma excursão é uma **Viagem** para um conjunto de **pessoas** (atributo do tipo ArrayList<Pessoa>) e que tem um custo de **alojamento** por pessoa (double **custop**).

b) Para a classe Excursao, defina um construtor que receba como parâmetros um objeto do tipo **Viagem** e um double com o custo do alojamento por pessoa.

c) Para a classe Excursao construa um método que calcule o custo total da excursão, incluindo o custo da viagem e do alojamento para cada passageiro. Deve ter em conta que o custo de alojamento varia com a idade da pessoa. Até aos 3 anos o alojamento é gratuito, para maiores de 3 anos e menos de 18 o preço é apenas 50%.

6 – Qual a diferença entre uma variável de classe e uma variável de instância? Dê exemplos.

7 – Qual a diferença entre sobrecarga (overloading) e sobreposição (overriding) de métodos? Dê exemplos.

8 – O que é uma classe genérica e para que serve?

9 – O que é uma interface funcional em java e para que serve?

Cotação: 1 – 3,75**; 2** – 0,75**; 3** – 2.0**; 4** – 0,75**; 5** – 3,25**; 6** – 0,75**; 7** – 0,75**; 8** – 0,5**; 9** – 0,5**;**

Classe java.time.LocalDate:

```
static LocalDate now ()
// Constrói um calendário com a data actual do seu computador.
static LocalDate of ( int ano, int mês, int dia)
//Constroi o objeto com a data dada nos parâmetros ano, mês e dia.
public boolean equals(Object obj)
public int compareTo (LocalDate d )
// compara dois calendários. Devolve 0 se a data que recebe a mensagem é
igual à data parâmetro. Devolve um valor <0 se a data que recebe a
mensagem é menor que a data parâmetro e devolve um valor >0 caso
contrário.
```

Classe java.util.ArrayList:

```
ArrayList() // construtor vazio, dimensão inicial zero.
boolean add(Object element)
// adiciona o elemento especificado ao final da lista
void add( int index, Object obj)
//insere o elemento especificado na posição index
Object remove(int index )//remove o elemento da posição index
boolean  remove( Object o)
//remove a primeira ocorrência do objecto dado como parâmetro
Object set (int position, Object obj )
//  substitui o elemento da posição position pelo elemento dado
Object get (int index)//devolve o elemento da posição index
void clear() // remove todos os elementos da lista
Object clone() // devolve uma cópia da lista
boolean contains(Object element)
// devolve true se a lista contém o elemento especificado
boolean equals ( Object obj)
// permite comparar duas listas
int indexOf(Object element)
 // procura o índice da 1ª ocorrência de elemento
boolean isEmpty() // verifica se a lista não tem componentes
int size() // devolve a dimensão actual
String toString ()
```