Exame de anos anteriores

Exame da época normal 2020/2021
(realizado no Moodle)

PARTE 1

Pergunta 1

- Qual será a saída resultante da execução deste programa?

```
1  class Base {
2      static int valor = 10;
3      Base() {
4          valor = 20;
5      }
6      static int getValor( String s) {
7          System.out.print( s + valor++);
8          return valor;
9      }
10 }
11 class A {
12     A() {
13         Base.getValor(", MsgAa:");
14     }
15     @Override
16     public boolean equals(Object obj) {
17         Base.getValor(", MsgAb:");
18         return obj instanceof A;
19     }
20 }
```

```
21 class B extends A {
22     B() {
23         Base.getValor(", MsgBb:");
24     }
25 }
26 class C extends B {
27     A aux;
28     C() {
29         Base.getValor(", MsgCa:");
30         aux = new B();
31         if (this.equals(aux))
32             Base.getValor(", MsgCb:");
33         Base.getValor(", MsgCc:");
34     }
35 }
36 public class Main {
37     public static void main(String[] args) {
38         System.out.print("Inicio_M");
39         new C();
40         System.out.println(", Fim_M");
41     }
42 }
```

Pergunta 1

- Qual será a saída resultante da execução deste programa?
  - Resposta:

```
Inicio_M, MsgAa:10, MsgBb:11, MsgCa:12, MsgAa:13, MsgBb:14, MsgAb:15, MsgCb:16, MsgCc:17, Fim_M
```

```java
class Base {
    static int valor = 10;
    Base() {
        valor = 20;
    }
    static int getValor( String s) {
        System.out.print( s + valor++);
        return valor;
    }
}
class A {
    A() {
        Base.getValor(", MsgAa:");
    }
    @Override
    public boolean equals(Object obj) {
        Base.getValor(", MsgAb:");
        return obj instanceof A;
    }
}
```

```java
class B extends A {
    B() {
        Base.getValor(", MsgBb:");
    }
}
class C extends B {
    A aux;
    C() {
        Base.getValor(", MsgCa:");
        aux = new B();
        if (this.equals(aux))
            Base.getValor(", MsgCb:");
        Base.getValor(", MsgCc:");
    }
}
public class Main {
    public static void main(String[] args) {
        System.out.print("Inicio_M");
        new C();
        System.out.println(", Fim_M");
    }
}
```

Pergunta 2

- Considere a classe `MinhaEscola`. Das seguintes opções, assinale aquelas que indicam métodos que, ao existirem na classe, permitem que seja concreta (não abstrata).
  - ler() e escrever()
  - ler(), escrever() e inscreverAluno()
  - inscreverAluno()
  - ler(), escrever(), cantar() e inscreverAluno()
  - cantar() e inscreverAluno()

```
1  interface Aprender{
2      void ler();
3      void escrever();
4      default void cantar(){
5          System.out.println(" musica ");
6      }
7  }
8  abstract class Escola implements Aprender{
9      abstract public void inscreverAluno();
10 }
11 class MinhaEscola extends Escola{
12 /* ... */
13 }
14 public class Main {
15     public static void main(String [] args){
16         MinhaEscola ob = new MinhaEscola();
17     }
18 }
```

Pergunta 2

- Considere a classe `MinhaEscola`. Das seguintes opções, assinale aquelas que indicam métodos que, ao existirem na classe, permitem que seja concreta (não abstrata).
  - ler() e escrever()
  - **ler(), escrever() e inscreverAluno()**
  - inscreverAluno()
  - **ler(), escrever(), cantar() e inscreverAluno()**
  - cantar() e inscreverAluno()

```
1  interface Aprender{
2      void ler();
3      void escrever();
4      default void cantar(){
5          System.out.println(" musica ");
6      }
7  }
8  abstract class Escola implements Aprender{
9      abstract public void inscreverAluno();
10 }
11 class MinhaEscola extends Escola{
12 /* ... */
13 }
14 public class Main {
15     public static void main(String [] args){
16         MinhaEscola ob = new MinhaEscola();
17     }
18 }
```

Pergunta 3

- Qual será a saída resultante da execução deste programa?

```
1  class EstaExcepcao extends Exception { }
2
3  public class Main {
4      static void f(int n) throws EstaExcepcao{
5          try{
6              if (n==1) throw new EstaExcepcao();
7              System.out.print(" sol ");
8          } catch( EstaExcepcao e){
9              System.out.print(" agua ");
10             throw e;
11         }   finally{
12             System.out.print(" terra ");
13         }
14         System.out.print(" fogo ");
15     }
16     public static void main( String[] args){
17         try{
18             f(1);
19             System.out.print(" lua ");
20         }
21         catch( EstaExcepcao e){
22             System.out.print(" ar ");
23         }
24         catch(Exception e){
25             System.out.print(" lago ");
26         }
27         System.out.print(" mar ");
28     }
29 }
```

Pergunta 3

- Qual será a saída resultante da execução deste programa?

- Resposta:

```
agua terra ar mar
```

```java
class EstaExcepcao extends Exception { }

public class Main {
    static void f(int n) throws EstaExcepcao{
        try{
            if (n==1) throw new EstaExcepcao();
            System.out.print(" sol ");
        } catch( EstaExcepcao e){
            System.out.print(" agua ");
            throw e;
        }   finally{
            System.out.print(" terra ");
        }
        System.out.print(" fogo ");
    }
    public static void main( String[] args){
        try{
            f(1);
            System.out.print(" lua ");
        }
        catch( EstaExcepcao e){
            System.out.print(" ar ");
        }
        catch(Exception e){
            System.out.print(" lago ");
        }
        System.out.print(" mar ");
    }
}
```

Pergunta 4

- Assinale quais das opções seguintes correspondem a linhas onde estão instruções que deveriam ser convenientemente substituídas para que não ocorressem erros de compilação.
  - As alíneas erradas têm cotação negativa para o cálculo da nota desta pergunta.
  - A cotação mínima desta pergunta é zero.

```
1  abstract class AnimalDeEstimacao{
2      private String especie;
3      public AnimalDeEstimacao(String especie){
4          this.especie = especie;
5      }
6      public String getEspecie() {
7          return especie;
8      }
9      public void setEspecie(String especie) {
10         this.especie = especie;
11     }
12 }
13 class Cao extends AnimalDeEstimacao {
14     public Cao(){  setEspecie("Cao");  }
15     public void ladrar(){ /*...*/}
16 }
17 class Gato extends AnimalDeEstimacao {
18     public Gato(){  setEspecie("Gato");  }
19     public void miar(){ /*...*/}
20 }
21 public class Main {
22     public static void main(String [] args){
23         List<AnimalDeEstimacao> animais = new ArrayList<>();
24         animais.add( new Cao());
25         animais.add( new Gato());
26         for( AnimalDeEstimacao a: animais){
27             if(a.getEspecie().equals("Cao")){
28                 a.ladrar();
29             }else if(a.getEspecie().equals("Gato")){
30                 a.miar();
31             }
32         }
33     }
34 }
```

Pergunta 4

- Assinale quais das opções seguintes correspondem a linhas onde estão instruções que deveriam ser convenientemente substituídas para que não ocorressem erros de compilação.
  - As alíneas erradas têm cotação negativa para o cálculo da nota desta pergunta.
  - A cotação mínima desta pergunta é zero.

- Resposta:

  14, 18, 28, 30

```
1  abstract class AnimalDeEstimacao{
2      private String especie;
3      public AnimalDeEstimacao(String especie){
4          this.especie = especie;
5      }
6      public String getEspecie() {
7          return especie;
8      }
9      public void setEspecie(String especie) {
10         this.especie = especie;
11     }
12 }
13 class Cao extends AnimalDeEstimacao {
14     public Cao(){  setEspecie("Cao");  }
15     public void ladrar(){ /*...*/}
16 }
17 class Gato extends AnimalDeEstimacao {
18     public Gato(){  setEspecie("Gato");  }
19     public void miar(){ /*...*/}
20 }
21 public class Main {
22     public static void main(String [] args){
23         List<AnimalDeEstimacao> animais = new ArrayList<>();
24         animais.add( new Cao());
25         animais.add( new Gato());
26         for( AnimalDeEstimacao a: animais){
27             if(a.getEspecie().equals("Cao")){
28                 a.ladrar();
29             }else if(a.getEspecie().equals("Gato")){
30                 a.miar();
31             }
32         }
33     }
34 }
```

Pergunta 5

- Qual será a saída resultante da execução deste programa?

```java
class Disciplina {
    private String nome;
    private int nAlunos;

    public Disciplina(String nome, int n) {
        this.nome = nome;
        this.nAlunos = n;
    }
    public void inscreveAluno(){ ++nAlunos; }
    public void desinscreveAluno(){ --nAlunos; }
    @Override
    public boolean equals(Object ob) {
        if (ob == null || getClass()!= ob.getClass())
            return false;
        return nome.equals(((Disciplina) ob).nome);
    }
    @Override
    public int hashCode() {  return this.nAlunos;  }
}
class Main {
    public static void main(String[] args) {
        Disciplina pa = new Disciplina("PA", 200);
        Disciplina poo = new Disciplina("POO", 300);

        Collection<Disciplina> coleccao1 = new ArrayList<>();
        coleccao1.add(pa);
        coleccao1.add(poo);

        Collection<Disciplina> coleccao2 = new HashSet<>();
        coleccao2.add(pa);
        coleccao2.add(poo);

        System.out.print(coleccao1.contains(poo) + " ");
        System.out.print(coleccao2.contains(poo) + " ");

        poo.inscreveAluno();
        System.out.print(coleccao1.contains(poo) + " ");
        System.out.print(coleccao2.contains(poo) + " ");
    }
}
```

Pergunta 5

- Qual será a saída resultante da execução deste programa?

- Resposta:

```
true true true false
```

```
1  class Disciplina {
2      private String nome;
3      private int nAlunos;
4
5      public Disciplina(String nome, int n) {
6          this.nome = nome;
7          this.nAlunos = n;
8      }
9      public void inscreveAluno(){ ++nAlunos; }
10     public void desinscreveAluno(){ --nAlunos; }
11     @Override
12     public boolean equals(Object ob) {
13         if (ob == null || getClass()!= ob.getClass())
14             return false;
15         return nome.equals(((Disciplina) ob).nome);
16     }
17     @Override
18     public int hashCode() {  return this.nAlunos;  }
19 }
20 class Main {
21     public static void main(String[] args) {
22         Disciplina pa = new Disciplina("PA", 200);
23         Disciplina poo = new Disciplina("POO", 300);
24
25         Collection<Disciplina> coleccao1 = new ArrayList<>();
26         coleccao1.add(pa);
27         coleccao1.add(poo);
28
29         Collection<Disciplina> coleccao2 = new HashSet<>();
30         coleccao2.add(pa);
31         coleccao2.add(poo);
32
33         System.out.print(coleccao1.contains(poo) + " ");
34         System.out.print(coleccao2.contains(poo) + " ");
35
36         poo.inscreveAluno();
37         System.out.print(coleccao1.contains(poo) + " ");
38         System.out.print(coleccao2.contains(poo) + " ");
39     }
40 }
```

Pergunta 6 a)

Considere um sistema de vendas online. Já existe implementada a funcionalidade para efetuar as operações básicas de suporte tal como obter item por nome, remover item do stock, por e tirar dinheiro na conta do cliente, por e tirar dinheiro da conta da empresa, fechar a venda, devolver item, etc., disponível em objetos do tipo *OnlineCommerce*.

```
public interface OnlineCommerce {
    int getItemID(String itemName);      // obtem código (ID) do item dado o nome. -1 se não existir
    boolean withdrawFromClient(int value, int clientIBAN);
        // retira dinheiro da conta do cliente. Retorna false (falha) se não tiver suficiente
    boolean depositIntoClient(int value, int clientIBAN);
        // repõe dinheiro da conta do cliente. Sucede bem sempre
    boolean addItemToCart(int clientID, int ItemID);          // O item é indicado por código
        // adiciona ao item carrinho de compras. Item fica logo reservado. Falha se não houver em stock
    boolean removeItemFromCart(int clientID, int ItemID);
        // remove item do carrinho de compras. Item fica "des-reservado". Falha se não estivesse no carrinho
    boolean emptyChart();    // esvazia carrinho de compras – (após finalizar todas as compras do cliente)
}
```

Pretende-se implementar a operação de ***adição de vários itens ao carrinho de compras de um cliente***. É ***fornecido um ArrayList*** com os nomes dos itens a serem adicionados e o ID do cliente. A concretização desta operação consiste em i) ***obter o ID de cada item*** e ii) para cada um, ***adicionar esses ID*** ao carrinho de compras do cliente. A operação pode ser desfeita apenas se tiver corrido bem. Se alguns dos passos/itens falhar, considera-se que toda a operação falhou.

a) Implemente esta operação através do ***padrão Command tal como lecionado nas aulas*** indicando a interface e a classe concreta para a concretização da operação pedida.

Pergunta 6 a)

```
interface ICommand {
    boolean execute();
    boolean undo();
}

class AddItemsToClient implements ICommand {
    OnlineCommerce commerce;
    ArrayList<String> prods;
    int clientID;

    public AddItemsToClient(OnlineCommerce commerce, int clientID, ArrayList<String> prods) {
        this.commerce = commerce;
        this.clientID = clientID;
        this.prods = prods;
    }
    @Override
    public boolean execute() {
        for(String p : prods) {
            int id = commerce.getItemID(p);
            if (id == -1 || !commerce.addItemToCart(clientID, id))
                return false; // numa resposta completa deveriam ser retirados os produtos adicionados,
                              // mas não era obrigatório no exame
        }
        return true;
    }
    @Override
    public boolean undo() {
        for(String p : prods) {
            int id = commerce.getItemID(p);
            if (id == -1 || !commerce.removeItemFromCart(clientID, id))
                return false;
        }
        return true;
    }
}
```

Pergunta 6 b)

b) Assuma que a variável ***manager*** tem uma referência válida para um ***CommandManager*** e a variável ***myShop*** tem uma referência válida para uma instância do tipo ***OnlineCommerce***. Usando a classe que fez na alínea a), escreva o código necessário para adicionar os itens "sabão", "escova", "esfregão" ao carrinho de compras do cliente cujo ID é 12321, de seguida adicionar os itens "alface", "cebola" ao carrinho do cliente com ID 50005, e depois desfazer a operação relativa ao cliente 50005.

Pergunta 6 b)

b) Assuma que a variável ***manager*** tem uma referência válida para um ***CommandManager*** e a variável ***myShop*** tem uma referência válida para uma instância do tipo ***OnlineCommerce***. Usando a classe que fez na alínea a), escreva o código necessário para adicionar os itens "sabão", "escova", "esfregão" ao carrinho de compras do cliente cujo ID é 12321, de seguida adicionar os itens "alface", "cebola" ao carrinho do cliente com ID 50005, e depois desfazer a operação relativa ao cliente 50005.

- Exemplo de resposta:

```
ArrayList<String> list1 = new ArrayList<>();
list1.add("sabão");list1.add("escova");list1.add("esfregão");

AddItemsToClient cmd1 = new AddItemsToClient(myShop, 12321, list1);
manager.invokeCommand(cmd1);

ArrayList<String> list2 = new ArrayList<>();
list2.add("alface");list2.add("cebola");

AddItemsToClient cmd2 = new AddItemsToClient(myShop, 50005, list2);
manager.invokeCommand(cmd2);
manager.undo();
```

PARTE 2

Descrição da pergunta sobre FSM

- Considere um tipo de relógio básico que permite apenas mostrar e acertar as horas, minutos e segundos:
  - O relógio é controlado por quatro botões: <on/off>, <set>, <+> e <->;
  - Em cada instante, pode estar em um dos cinco modos seguintes: “Desligado”, “Display”, “Acerto das Horas”, “Acerto dos Minutos” e “Acerto dos Segundos”;
  - O botão <on/off> permite ligar o relógio caso esteja desligado, passando este para o modo “Display”, ou desligá-lo em qualquer um dos restantes modos;
  - O botão <set> tem por efeito mudar o modo de acordo com a seguinte sequência: “Display” -> “Acerto das Horas” -> “Acerto dos Minutos” -> “Acerto dos Segundos” -> “Display” -> “Acerto de Horas” -> …
  - Nos modos de acerto, os botões <+> e <->, respetivamente, incrementam e decrementam de uma unidade a componente correspondente (horas, minutos ou segundos).
- Considere que se pretende implementar, em linguagem Java, este tipo de relógio básico sob a forma de uma máquina de estados orientada a objetos. Recorra, para o efeito, ao padrão estudado nas aulas e aplicado ao trabalho prático para que um relógio básico seja uma instância de uma classe que, entre outros atributos, inclua o seu estado atual e tire partido do polimorfismo ao nível dos estados.

Pergunta 1 e 2

- Idealize um diagrama que represente a máquina de estados `BasicClock` de forma adequada, com atribuição de nomes aos estados e às transições constantes da seguinte lista: `Off`, `Display`, `SetHours`, `SetMinutes`, `SetSeconds`, `Set`, `OnOff`, `Increment` e `Decrement`.

- Implemente uma interface `IStates` apropriada à máquina de estados `BasicClock`.

Pergunta 3

- Desenvolva uma classe `Adapter` que implemente a interface `IStates` e a partir da qual derivam todos os estados da máquina de estados `BasicClock`. Esta classe deve ter acesso a um objeto do tipo `ClockData`.

```
public class ClockData {
    private int h, m, s;

    public ClockData(int h, int m, int s) {
        this.h = h;
        this.m = m;
        this.s = s;
    }

    public int getH() {return h;}

    public void setH(int h) {
        if(h>=0 && h<=23)
            this.h = h;
    }

    public int getM() {return m;}

    public void setM(int m) {
        if(m>=0 && m<=59)
            this.m = m;
    }

    public int getS() {return s;}

    public void setS(int s) {
        if(s>=0 && s<=59)
            this.s = s;
    }

    @Override
    public String toString() {
        return  String.format("%02d:%02d:%02d", h, m, s);
    }

}
```

Pergunta 4 e 5

- Implemente uma classe que represente o estado em que um relógio básico se encontra desligado.

- Implemente uma classe que represente o estado em que é possível acertar os minutos de um relógio básico.

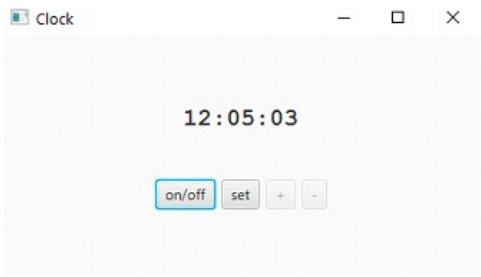
Descrição da pergunta sobre JavaFX

- Considere que se pretende desenvolver, recorrendo ao JavaFX, uma interface do utilizador em modo gráfico (GUI) para um relógio básico.

- Assuma, igualmente, uma abordagem de notificações assíncronas semelhante à aplicada durante as aulas e no trabalho prático, ou seja, baseada na utilização da classe `java.beans.PropertyChangeSupport` e da interface `java.beans.PropertyChangeListener`.

- As figuras ao lado exemplificam a GUI pretendida.

a) Desligado



b) Ligado

c) Acerto dos minutos

Pergunta 6

- Complete o código seguinte para que a classe `ObservableBasicClock`, referenciada pela GUI, atue efetivamente como modelo observável:

```
[Altere este código]
public class ObservableBasicClock {
    public static final String UPDATE_PROPERTY = "update";
    private BasicClock basicClock;
    PropertyChangeSupport pCSupport;

    public ObservableBasicClock(int h, int m, int s) {
        basicClock = new BasicClock(h, m, s);
        pCSupport = new PropertyChangeSupport(this);
    }

    public void addPropertyChangeListener(String property, PropertyChangeListener listener) {
        pCSupport.addPropertyChangeListener(property, listener);
    }

    public int getH() {return basicClock.getH();}
    public int getM() {return basicClock.getM();}
    public int getS() {return basicClock.getS();}

    public ESituation getSituation() {return basicClock.getSituation();}

    public void onOff() {basicClock.onOff();}
    public void set() {basicClock.set();}
    public void increment() {basicClock.increment();}
    public void decrement() {basicClock.decrement();}

    @Override
    public String toString() {return basicClock.toString();}
}
```

Pergunta 7

- Complete, onde e apenas onde é solicitado, o código da classe `ClockPane`, responsável por representar o relógio referenciado e permitir a interação do utilizador.

```
public class ClockPane extends BorderPane {
    private ObservableBasicClock obs;

    private Label display;
    private Button onOff, set, inc, dec;

    public ClockPane(ObservableBasicClock obs) {
        this.obs = obs;

         /*Registar um property change listener em obs e cujo método
         propertyChange() invoque apenas o metodo update()*/

        /* A */

        display = new Label();

        onOff = new Button("on/off");
        set = new Button("set");
        inc = new Button("+");
        dec = new Button ("-");

        arrangeComponents();
        setEventHandlers();
        update();
    }

    private void arrangeComponents() {

        HBox buttonBox = new HBox();
        buttonBox.getChildren().addAll(onOff, set, inc, dec);

        VBox vBox = new VBox();
        vBox.getChildren().addAll(display, hBox);

        setCenter(vBox);
    }

    private void setEventHandlers() {

        //Definir a resposta a acoes dos botoes onOff, set, inc e dec.
        /* B */
    }

    private void update() {

        ESituation situation = obs.getSituation();

        //Definir o texto da label display.
        /* C */

         //Definir o estado de ativiacao (disable/enable) dos botões set, inc e dec.
        /* D */
    }
}
```