PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

PCS-2302 / PCS-2024
Lab. de Fundamentos de Eng. de Computação

Aula 04

Fundamentos da Programação Orientada a Objetos (Parte 2)

Professores:
Anarosa Alves Franco Brandão (PCS2302)
Jaime Simão Sichman (PCS 2302)
Reginaldo Arakaki (PCS 2024)
Ricardo Luís de Azevedo da Rocha (PCS 2024)

Monitores: Diego Queiroz e Tiago Matos

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Roteiro

- Elementos básicos da POO – Parte 2
  - Variáveis e objetos
  - Criação de objetos
  - Mutabilidade
  - Tratamento de Exceções
- Parte Experimental
  - Implementação do simulador MVN (parte 2)
    - Classe **`Registradores`**
    - Classe **`UnidadeControle`**

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Variáveis e objetos (1)

- Variáveis de instância: são as variáveis que implementam a representação dos dados de uma abstração de dados. São os atributos da classe (Java).

```
public class Word {

 private Bits8 byte0;
 private Bits8 byte1;

 public Word() {
  byte0 = new Bits8(0);
  byte1 = new Bits8(0);
 }

 public Word(Bits8 byt0, Bits8 byt1) {
  byte0 = byt0.makeCopy();
  byte1 = byt1.makeCopy();
 }

 public void setHex(String hexa) {
  int tamanho = hexa.length();
  StringBuffer buf = new StringBuffer(hexa);

  if (tamanho < 4) {
   buf.reverse();
   for (int i = 0; i < (4 - tamanho); ++i)
    buf.append('0');
   hexa = buf.reverse().toString();
  }
            :
            :
```

Exemplo: Classe **Word** da MVN dada na aula 3

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Variáveis e objetos (2)

- Variáveis locais, tais como as declaradas nos métodos, têm seu espaço alocado na chamada do método e desalocado quando do retorno do método.

```
public class Word {

 private Bits8 byte0;
 private Bits8 byte1;

 public Word() {
  byte0 = new Bits8(0);
  byte1 = new Bits8(0);
 }

 public Word(Bits8 byt0, Bits8 byt1) {
  byte0 = byt0.makeCopy();
  byte1 = byt1.makeCopy();
 }

 public void setHex(String hexa) {
  int tamanho = hexa.length();
  StringBuffer buf = new StringBuffer(hexa);

  if (tamanho < 4) {
   buf.reverse();
   for (int i = 0; i < (4 - tamanho); ++i)
    buf.append('0');
   hexa = buf.reverse().toString();
  }
            :
            :
```

Exemplo: Classe **Word** da MVN dada na aula 3

Variáveis e objetos (3)

- As variáveis dos tipos primitivos contêm arranjos de bits representando os valores das variáveis (8, -45.9, 'z', etc.).

```
public class Word {

 private Bits8 byte0;
 private Bits8 byte1;

 public Word() {
  byte0 = new Bits8(0);
  byte1 = new Bits8(0);
 }

 public Word(Bits8 byt0, Bits8 byt1) {
  byte0 = byt0.makeCopy();
  byte1 = byt1.makeCopy();
 }

 public void setHex(String hexa) {
  int tamanho = hexa.length();
  StringBuffer buf = new StringBuffer(hexa);

  if (tamanho < 4) {
   buf.reverse();
   for(int i = 0; i < (4 - tamanho); ++i)
    buf.append('0');
   hexa = buf.reverse().toString();
  }
          .
          .
          .
```

Exemplo: Classe **Word** da MVN dada na aula 3

Variáveis e objetos (4)

- Não há algo como uma variável cujo valor represente um objeto, mas sim uma variável cujo valor é uma referência para um objeto.
- O valor da variável é um arranjo de bits representando um meio de acesso a um objeto específico residente no *heap*.
- Os objetos são criados no *heap* pelo operador **new**.

```
public class Word {

 private Bits8 byte0;
 private Bits8 byte1;

 public Word() {
  byte0 = new Bits8(0);
  byte1 = new Bits8(0);
 }
          .
          .
          .
```

Exemplo: Classe **Word** da MVN dada na aula 3

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Variáveis e objetos (5)

- Em Java, as variáveis de instância têm um valor inicial *default*: tipos primitivos numéricos (inclusive char) têm zero, booleanos têm falso e variáveis de referência a objeto têm nulo.

- As variáveis locais devem ser iniciadas antes de seu uso. Se o compilador não puder provar que uma variável foi iniciada antes de seu uso, causará um erro de compilação.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Criação de objetos (1)

- Objetos são criados via operador **new** e um construtor como operando. O **new** aloca memória para o objeto no *heap*, iniciando-a com valores *default* ou designados pelo construtor

```
public class Simples {
  // Campos (atributos)
  private int i, j;
  private int[] a = {1, 2, 3, 4};
  private int[] b;
  private String s, t;
  // Construtor default
  Simples() {
    i = 3;
    b = new int[3];
    s = "abcd";
    t = null;
  }
  public static void main(String[] args) {
    // Cria um objeto obj1
    Simples obj1 = new Simples();
  }
}
```

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

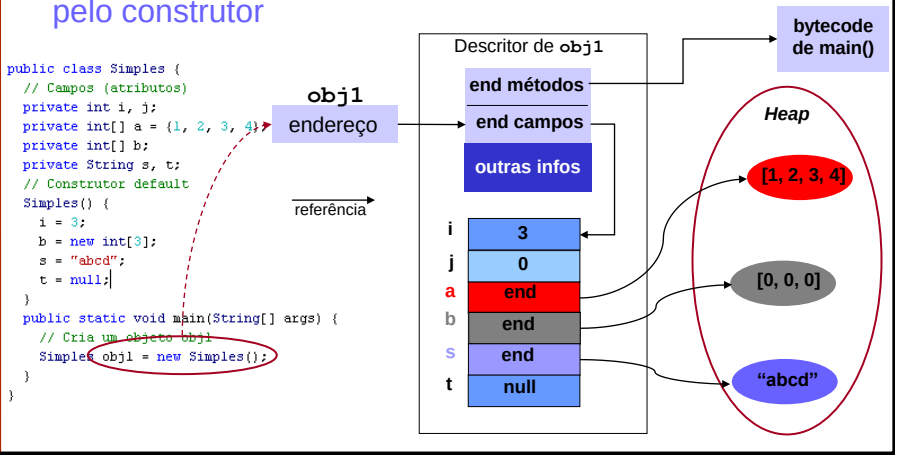
Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Criação de objetos (1)

- Objetos são criados via operador **`new`** e um construtor como operando. O **`new`** aloca memória para o objeto no *heap*, iniciando-a com valores *default* ou designados pelo construtor

```
public class Simples {
  // Campos (atributos)
  private int i, j;
  private int[] a = {1, 2, 3, 4};
  private int[] b;
  private String s, t;
  // Construtor default
  Simples() {
    i = 3;
    b = new int[3];
    s = "abcd";
    t = null;
  }
  public static void main(String[] args) {
    // Cria um objeto objl
    Simples objl = new Simples();
  }
}
```



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Criação de objetos (2)

```
public class Simples {
  // Campos (atributos)
  private int i, j;
  private int[] a = {1, 2, 3, 4};
  private int[] b;
  private String s, t;
  // Construtor default
  Simples() {
    i = 3;
    b = new int[3];
    s = "abcd";
    t = null;
  }
  public static void main(String[] args) {
    Simples objl = new Simples(); // Instancia um objeto objl
    objl.mostrarCampos(); // Invoca métodos de objl
    objl.metodo();
    objl.mostrarCampos();
  }
  private void metodo() { // Visibilidade apenas interna do objeto
    j = i;     b = a;     t = s;
  }
  public void mostrarCampos() {
    System.out.println("i = " + i + "\tj = " + j);
    System.out.println("a = " + a + "\tb = " + b);
    System.out.println("s = " + s + "\tt = " + t);
  }
} // fim simples
```

Ponto de entrada do programa.

Método estático aplica-se à classe em que é declarado, podendo realizar tarefas independentes do conteúdo de outros objetos. Também conhecido como método da classe.

Por isso usa método `mostrarCampos()` ao invés de invocar diretamente `System.out.println(...)`

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Criação de objetos (3)

- Semântica de chamada de método:
  - Uma chamada **`obj.metodo()`**, inicialmente avalia **`obj`** para obter o objeto cujo método está sendo chamado.
  - Em seguida, avalia (Java: da esquerda para a direita) as expressões dos argumentos para obter os valores dos parâmetros reais.
  - Logo após, os parâmetros reais são copiados nos parâmetros formais do método – chamada por valor.
    - Copia-se o arranjo de bits (valor). Ele é adequadamente interpretado como valor de um tipo primitivo ou como valor de uma referência para um objeto.
  - Finalmente, o controle é despachado para o método invocado.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

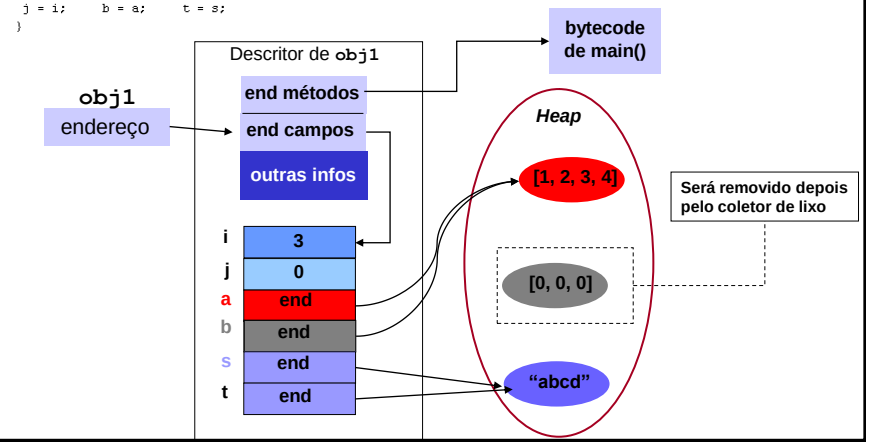
Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Criação de objetos (4)

- Após a invocação de **`metodo()`**.

```
public static void main(String[] args) {
  Simples obj1 = new Simples(); // Instancia um objeto obj1
  obj1.mostrarCampos(); // Invoca métodos de obj1
  obj1.metodo();
  obj1.mostrarCampos();
}
private void metodo() { // Visibilidade apenas interna do objeto
  j = i;     b = a;     t = s;
 }
```



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Mutabilidade (1)

- Um objeto é mutável se seu estado puder mudar. Ex., um array é mutável
- Um objeto é imutável se seu estado nunca muda. Ex. Strings são imutáveis

String **t** é imutável: um outro objeto é criado na memória para armazenar o valor da soma **t+"e"**

```
public class Simples {
  // Campos (atributos)
  private int i, j;
  private int[] a = {1, 2, 3, 4};
  private int[] b;
  private String s, t;
  // Construtor default
  Simples() {
    i = 3;     b = new int[3];
    s = "abcd";    t = null;
  }
  public static void main(String[] args) {
    Simples obj1 = new Simples();
    obj1.mostrarCampos();
    obj1.metodo();
    obj1.mostrarCampos();
  }
  private void metodo() {
  // Visibilidade apenas interna do objeto
    j = i;     b = a;     t = s;
    t = t + "e" // string t é imutável
    }

  public void mostrarCampos() {
    System.out.println("i = " + i + "\tj = " + j);
    System.out.println("a = " + a + "\tb = " + b);
    System.out.println("s = " + s + "\tt = " + t);
  }
} // fim simples
```

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

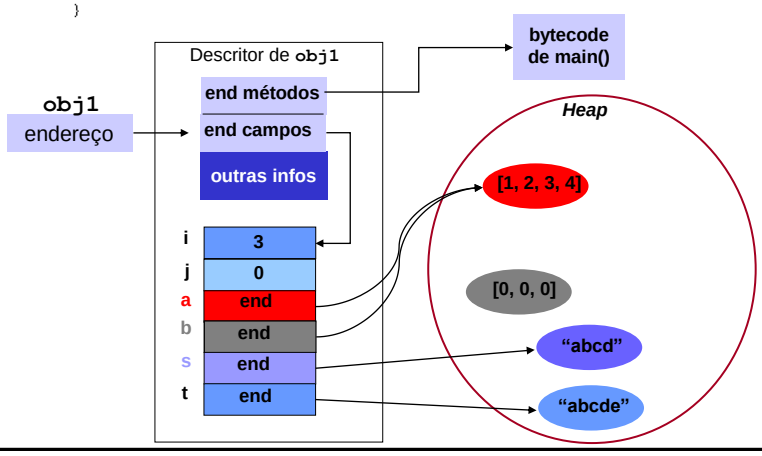
Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Mutabilidade (2)

- Após a invocação de **metodo().**

```
private void metodo() { // Visibilidade apenas interna do objeto
  j = i;
  b = a;
  t = s;
  t = t + "e"; // String t é imutável
}
```



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Mutabilidade (3)

- Por exemplo:

```
String op = "Concatenacao";
```

- Para alterar caracteres individuais na *string* **op** tenho que fazer:

```
op = op.substring(0,8) + "ção";
```

  – Mudo o conteúdo da variável **op**, fazendo-a referenciar uma outra **String**.

- Embora seja ineficiente gerar uma nova **String**, o seu caráter imutável tem uma vantagem: as *strings* são compartilhadas. Quando se copia uma variável **String**, tanto a original quanto a cópia compartilham os mesmos caracteres.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Mutabilidade (4)

- Um objeto é compartilhado por duas variáveis se ele puder ser referenciado por ambas

- Se um objeto mutável for compartilhado por duas variáveis, as modificações feitas através de uma variável serão visíveis quando o objeto for usado pela outra variável

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Rudimentos de exceções (1)

- Um método é uma abstração procedimental que mapeia argumentos (domínio) em um resultado (contra-domínio).
- O uso de procedimentos parciais requer uma espécie de contrato em que o código que invoca a chamada deve assegurar que os argumentos estejam no subconjunto permitido do domínio.
- Problemas na execução do contrato podem gerar erros, que podem ser contornados, por exemplo, se o código chamado ignorar os argumentos que não pertençam a este subconjunto.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Rudimentos de exceções (2)

- Uma exceção é uma ocorrência não esperada na execução de um programa. Indica a existência de erro, que pode ser tratado através da introdução de códigos de recuperação no corpo do programa, conhecidos como rudimentos de exceção.
- Na disciplina não será enfatizado o uso sistemático do mecanismo de exceção em Java. No entanto, o uso de vários objetos da biblioteca Java requerem o tratamento de exceções.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Rudimentos de exceções (3)

- Um programa robusto é o que continua a se comportar razoavelmente mesmo na presença de erros, no mínimo fornecendo uma degradação controlada.
- O uso de procedimentos totais aumenta a robustez: os comportamentos deveriam ser definidos para todas a entradas do domínio, com notificação do problema ocorrido para o código que o invoca. Como este pode ser notificado?

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

v. 1.4 set 2009

Rudimentos de exceções (4)

- Estratégia clássica para notificação de erro:
  - retornar um código de erro definido.
- Problema:
  - existem situações em que todos valores de retorno, ou quase todos, são resultados plausíveis. Além disso, a situação anômala deve ser bem distinguida das demais, de tal modo que os usuários não possam ignorá-la.

Rudimentos de exceções (5)

- O mecanismo de exceção fornece uma solução adequada: o método termina normalmente, retornando um resultado pertencente a seu contradomínio, ou termina excepcionalmente.
  - Como podem haver diferentes espécies de términos, pode-se ter tipos de exceção diferentes.
- A especificação de um método que pode terminar excepcionalmente tem a seguinte cláusula adicionada à sua assinatura:

```
throws <lista_de_tipos>
```


Rudimentos de exceções (6)

- Ex. Na classe **`Memoria`** da MVN tem-se:

```
public boolean loadArqTexto(String fileName)
    throws IOException { }
```

  Aqui, o método pode lançar um objeto do tipo **`IOException`** no caso de falha de E/S. Um método pode lançar mais de um tipo de exceção.
  - No código, a criação de um *buffer* de leitura para o arquivo texto requer o tratamento de uma exceção do tipo **`IOException`**. No caso, a sua verificação é obrigatória.
- Maiores detalhes e, sobretudo, como projetar e programar utilizando exceções podem ser encontrados, por exemplo, nas referências bibliográficas citadas.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Rudimentos de exceções (7)

- As exceções não devem ser usadas para situações esperadas simples.
  - Por ex., atingir o final de um *stream* de entrada é um comportamento esperado. Retornar um código de final de entrada é razoável e identifica a situação.
- Por outro lado, continuar a ler após o final da entrada não é uma situação esperada. Significa que o programa não percebeu o final e está tentando fazer algo que não deveria tentar fazer. Neste caso uma exceção, por ex., `ReadPastEndException` é interessante.
- A mensagem é: não abuse das exceções como meio de notificar situações esperadas.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Exercícios

TYGXXA04E01_09

- Esta segunda parte da implementação do simulador MVN tem por finalidade a sedimentação dos conceitos elementares da programação OO em Java.
- O simulador deve carregar um programa MVN escrito em arquivo texto para a memória e executar o programa, segundo as especificações apresentadas na Aula 2 e reproduzidas adiante com os ajustes adequados para o simulador utilizado na disciplina.
- A implementação estende o código produzido na Parte 1 da implementação da MVN.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

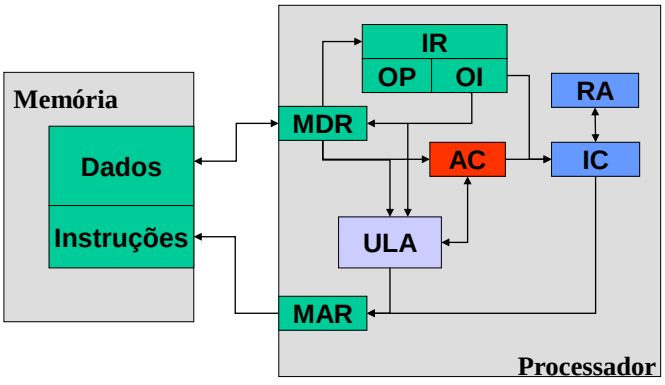
Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (1)

- Pausa para pensar: qual a estratégia de decomposição para resolver o problema? Na aula 2 vimos que a arquitetura a simular era:



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

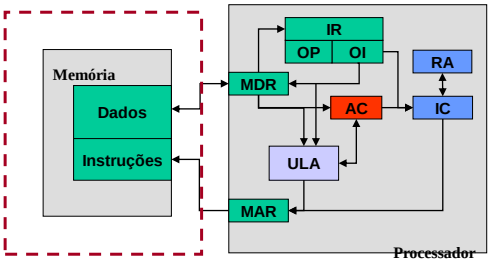
Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (2a)

- Quais os candidatos a abstrações de dados, ou melhor, quais os candidatos a tipos de objetos para uma solução?
  - **Memória**: visto na aula 3.



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

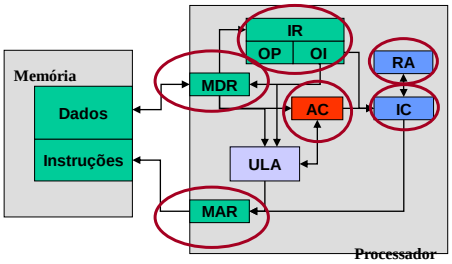
Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (2b)

- Quais os candidatos a abstrações de dados, ou melhor, quais os candidatos a tipos de objetos para uma solução?
  - **Memória**: visto na aula 3.
  - **Registradores**: Acumulador, MAR, MDR, IC, etc.



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

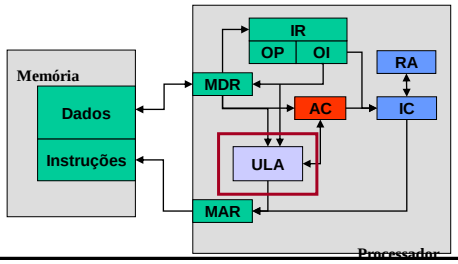
Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (2c)

- Quais os candidatos a abstrações de dados, ou melhor, quais os candidatos a tipos de objetos para uma solução?
  - **Memória**: visto na aula 3.
  - **Registradores**: Acumulador, MAR, MDR, IC, etc.
  - Unidade Lógica Aritmética, ou **Unidade de Controle**: responsável pela realização do ciclo FDE, durante o qual interage com a memória e os registradores.



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

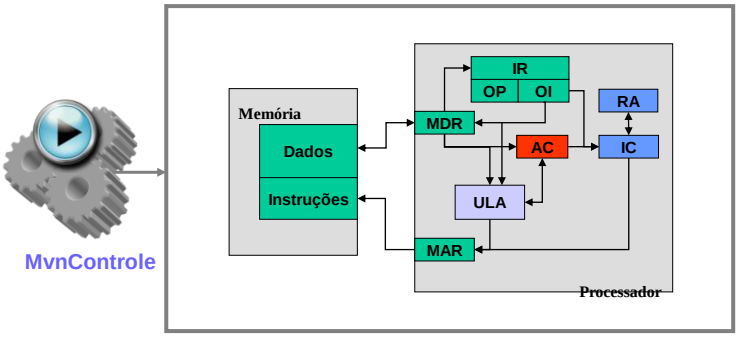
Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (3)

- Além destas abstrações, temos outra abstração interessante:
  - o **MvnControle**, responsável pela operação geral do simulador.



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

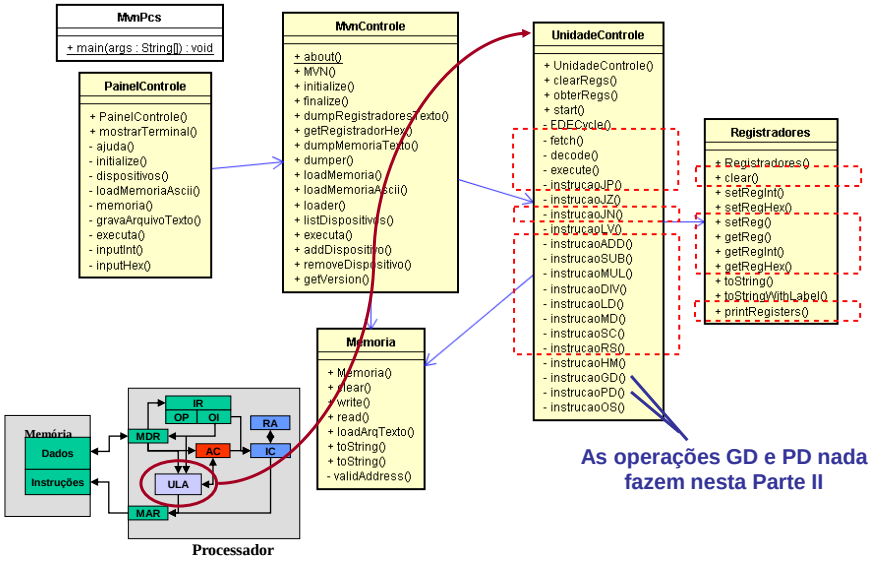
Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (4)



PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (5)

- Vocês receberão o código fonte incompleto, com especificações parciais de alguns métodos e a implementação de alguns métodos.
  - Implementar o simulador MVN de acordo com o diagrama de classes, codificando os métodos especificados e acrescentando métodos faltantes e/ou campos (atributos), com suas especificações e códigos, para as classes **`Registradores`** e **`UnidadeControle`**.
  - A implementação deverá **obrigatoriamente** estender o código da Parte 1 da MVN produzido anteriormente pelo grupo.
  - Obrigatório: **Especificar todos os métodos no estilo apresentado.**

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (6)

- **`PainelControle`**, **`Bits8`** e **`Word`**
  - Fornecidos completos. Painel apropriado para a Parte 2, mas verificar se há necessidade de complementação.
- Registradores
  - Fornecidos: **`setRegInt()`**, **`setRegHex()`**, **`toString()`**.
- UnidadeControle
  - Fornecidos: **`clearRegs()`**, **`obterRegs()`**, **`start()`**, **`FDECycle()`**, dica para **`execute()`**, **`instrucaoJZ()`**, **`instrucaoLV()`**, **`instrucaoOS()`**.
    - As instruções de E/S não farão nada na Parte II.
    - O método **`FDECycle()`** controla o ciclo de obtenção da instrução (*fetch*), decodificação da instrução (*decode*) e execução da instrução (*execute*).
    - Atenção para as especificações das instruções 0x000A (desvio para subprograma ou chamada de sub-rotina) e 0x000B (retorno de subprograma).

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Implementação da MVN - Parte 2 (7)

- Todos os arquivos-fonte vêm comentados, incluindo as especificações dos métodos. No entanto, muitos métodos não estão especificados no estilo da abstração procedimental.

  Colocá-los todos neste estilo.
- Instruções e esclarecimentos adicionais em sala de aula.

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha
© 2009

Aula 4:

Fundamentos Prog. Orientada Objetos (2)

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Algumas dicas

- Na classe `UnidadeControle`, devem-se manipular os registradores segundo os comentários do código.
- Uma das responsabilidades do método *execute* é invocar o método "instrução" adequado.
- Caso haja qualquer dúvida sobre uma classe do Java veja a documentação em: http://java.sun.com/javase/6/docs/api/

PCS 2302/2024
Laboratório de Fundamentos da Eng de Computação

Professores:
Anarosa A.F. Brandão
Jaime S. Sichman
Reginaldo Arakaki
Ricardo L.A. Rocha

Autores:
Anarosa A.F. Brandão
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Bibliografia Complementar

Budd, T. *An Introduction to Object-Oriented Programming*. 3a. Ed. Addison Wesley, 2001.

Liskov, B.; Guttag, J. *Program Development in Java*. Addison-Wesley, 2001.