1 Introdução

1.1 Terminologia

Definir os seguites termos:

- padrões elementares
- verificação de programas com casos de teste
- depuração de programas
  - sessão de depuração de programas
  - sistema de depuração automática de programas (diagnóstico de programas)
  - discriminação de hipóteses
  - intenções
  - componentes abstratos

2 Padrões Elementares

Os padrões elementares são soluções para problemas computacionais que ocorrem no cotidiano de aprendizes de programação. O objetivo desses padrões é permitir que o aprendiz de programação consiga responder questões como:

- Qual o problema que deve ser resolvido?
- Como esse problema pode ser resolvido?
- Qual é a melhor escolha em uma determinada situação?
- Como os vários elementos envolvidos na solução interagem?

Dessa forma, podemos considerar que os padrões elementares constituem uma forma do aluno aprender estratégias de resolução de problemas utilizando uma linguagem de programação.
Um professor experiente pode definir um conjunto de padrões que ele considera interessante que seus alunos aprendam e fornece a seus alunos, o que chamamos de *catálogo de padrões*. Um catálogo de padrões é um conjunto de padrões documentados e que podem ser utilizados por outras pessoas, nesse caso, os alunos. A comunidade envolvida com padrões possui algumas formas padronizadas para documentar os padrões, mas isso não é uma regra e pode ser feito de acordo com as necessidades ou o contexto.
Você pode encontrar mais detalhes sobre padrões em geral e padrões elementares em: http://csis.pace.edu/ bergin/, no documento: *Slides for a Patterns Short Course*.
No texto a seguir, apresentamos alguns padrão elementares que consideramos úteis para alguns problemas interessantes que os aprendizes de programação enfrentam.

2.1 Notação dos Padrõs Elementares

Nesse texto, definimos a nossa própria forma de documentar os padrões. Essa documentação é simples e objetiva para que os alunos possam encontrar rapidamente o que procuram e também para que possam compreender com facilidade o seu uso. Os seguintes itens compõem a nossa documentação dos padrões elementares:

- *Objetivo*: descreve de forma sucinta a função principal do padrão elementar.
- *Sintaxe*: mostra como o padrão é construído.
- *Semântica*: mostra os detalhes do funcionamento do padrão.
- *Exemplo de uso*: Apresenta um exemplo de uso do padrão, através da construção de um programa que resolva um determinado problema.
  - *Apresentação do Problema*: descrição do problema.
  - *Análise do Problema*: faz uma análise do problema e apresenta uma estratégia para resolvê-lo. Também é apresentada uma simulação simples usando a estratégia apresentada.
  - *Solução para o Problema*: mostra o programa solução para o problema.

3 Catálogo de Padrões Elementares

3.1 Repetição contada

Objetivo

Executar uma seqüência de ações por um número de vezes previamente conhecido.

Sintaxe

```
for (<inicialização-controle>; <condição>; <atualização-controle>) {
    <ações>
}
```

Semântica

Para utilizar o padrão, deve haver uma variável de controle reponsável por contar o número de vezes que as <ações> são executadas (*iterações*). Essa variável é chamada de *variável de controle*. Antes da primeira iteração, a variável de controle deve ser inicializada e em seguida, a <condição> é avaliada. Caso a <condição> seja verdadeira, então a iteração será executada. Essa <condição> deve envolver o valor da variável de controle. Após o término da iteração é feito a atualização do valor da variável de controle e a <condição> é novamente avaliada e executada a iteração caso a <condição> continue verdadeira. Esse processo é executado até que a condição se torne falsa, caso no qual o padrão de repetição contada é finalizado.

Exemplo de uso

Problema da nota mais alta . Um professor gostaria de saber qual a nota de prova mais alta entre os alunos da sua turma. Para ajudá-lo nessa tarefa, será necessário construir um programa para encontrar essa nota mais alta em uma turma. O programa deverá receber como entrada um número inteiro n que representa a quantidade de alunos da turma, seguido de n inteiros que representam a nota de cada um dos alunos. Após a execução, o programa deve devolver a nota mais alta da turma (observe que vários alunos podem ter tirado a mesma nota, inclusive essa nota mais alta).

Análise do problema Para compreendermos melhor o problema, vamos fazer uma simulação da forma como um professor faria, caso ele quisesse descobrir qual a nota mais alta de uma turma e não tivesse um programa a sua disposição, somente uma pilha de provas corrigidas. Vamos fazer essa simulação para uma turma de exemplo contendo 5 alunos, a suas notas de provas foram:

#Aluno	**1**	**2**	**3**	**4**	**5**
Nota	6	2	6	8	7

Ao verificar a nota do primeiro aluno, o professor assume que essa é a maior nota até o momento (isso porque nenhuma nota foi vista anteriormente). O professor então verifica a nota do aluno 2 (nota 2) e, mentalmente, compara com a maior nota vista até o momento, que é a nota 6. Assim, ele reconhece que a maior nota ainda é a nota 6. Ao ver a nota do aluno 3 (nota 6) o professor repara que é igual à maior nota até o momento e simplesmente passa para a próxima nota. O professor verifica que a nota do aluno 4 é 8, e como $8 > 6$, então a maior nota passa a ser 8. O valor da última nota é 7, mas como a maior nota vista é 8, o professor ainda mantém essa nota como sendo a maior. Como não há mais notas para serem verificadas, o professor conclui que a maior nota da turma é 8.
Um programa que resolve o *Prolema da nota mais alta*, deve, de certa forma, simular essa estratégia usada na simulação. Para tanto, vamos explorar os detalhes do problema.
Em primeiro lugar, observe que o número de alunos da turma é informado. Dessa forma, também sabemos quantas serão as notas. Seguinte a estratégia usada pelo professor, ele verificou cada uma das notas e comparou com a maior nota ele tinha visto até o momento. Se a nota que ele está verificando em um determinado momento (vamos chamá-la de nota atual) for maior que a maior nota já vista, então essa nota atual deve ser considerada como sendo

a maior já vista. Caso contrário, aquela considerada como sendo a maior nota vista, continuará sendo a maior. Esse tipo de comportamento pode ser escrito em um programa utilizando o padrão de Repetição Contada. No início do programa é feita a leitura do inteiro n, representando os alunos da turma e, em seguida, é executado o laço do padrão para ler cada uma das notas. Para cada nota lida, deve ser verificado se ela é maior do que a maior nota já lida até o momento. Note que para fazer esse processo o programa precisa armazenar somente a maior nota que foi lida, para que possa ser comparada com a última nota lida. Se a última nota lida é maior que a maior nota até esse momento, então essa última é deve ser considerada a maior nota. Caso contrário aquela que era a maior nota ainda continuará sendo a maior. Observe que é necessário fazer um tratamento especial para o caso da primeira nota lida, na qual ainda não existe essa maior nota para fazer a comparação. Nesse caso, podemos assumir que a maior nota até o momento tem o valor 0. Como não são consideradas notas menores que 0, é garantido que a maior nota será pelo menos a própria nota 0 (situação em que toda a turma tirou 0 na prova).
Apresentamos a seguir um programa que resolve esse problema.

Solução para o Problema da nota mais alta .

```
01  public static void main(String[] args) {
02    int i;     /* contador para o número de alunos */
03    int n;     /* armazena o número de alunos */
04    int maior; /* armazena a maior nota vista até o momento */
05    int atual; /* armazena a nota do i-ésimo aluno */

06    maior = 0;
07    n = readInt();

08    for (i = 0; i < n; i = i + 1) {
09      atual = readInt();
10      if (atual > maior)
11        maior = atual;
12    }

13    writeInt("A maior nota e'", maior);
14  }
```

3.2 Repetição com Sentinela

Objetivo

Executar uma seqüência de ações por um número indeterminado de vezes até que seja informado um valor especial indicando não há mais valores para

serem processados.

Sintaxe

```
ler um valor para <varDado>
while(<vardDado> != <sentinela>) {
  <ações>
  ler um novo valor para <varDado>
}
```

Semântica

Para utilizar o padrão, deve haver uma variável no qual terá seu valor atualizado a cada iteração do laço. O valor dessa variável é sempre testado em relação a um valor esperado, chamado de sentinela e representado por <sentinela>. O sentinela é um valor inválido que não deve ser processado, mas serve como um indicador para informar que os valores que deveriam ser fornecidos como entrada terminaram. Antes da primeira iteração, o valor de <varDado> é lido e essa valor é comparado com o valor de <sentinela> no teste de condição do laço. No caso dos valores serem iguais, o laço é finalizado e o programa segue sua execução após o final de bloco do laço. Caso contrário, as ações no bloco do laço são executadas. Entre as ações executadas no laço está uma ação de leitura do valor de <varDados> (geralmente a última ação) que servirá para o teste de condição e possível execução da próxima iteração do laço.

Exemplo de uso

Problema do dado . Uma pessoa gostaria saber a soma total obtida numa seqüência de lançamento de dados, e também o número de vezes que cada valor foi obtido no lançamento. Para auxiliar essa pessoa, você deve construir programa que recebe como entrada essa seqüência de valores obtidos nos lançamentos dos dados e devolver a soma total obtida nos lançamentos e o número de vezes que cada face do dado foi obtida. Observe que valores válidos na entrada são números naturais entre 1 e 6. Como o número de lançamentos não é fornecido de antemão, o seu programa deverá finalizar o processamento quando o valor 0 for obtido como entrada.

Análise do problema Como o número de lançamentos do dado não é fornecido, devemos resolver esse problema processando todos número naturais obtidos na entrada, que sejam entre 1 e 6, e finalizando o programa quando o número obtido for 0. O processamento será feito somando-se o valor obtido em uma variável de soma geral e para cada valor obtido na entrada,

devemos somar uma unidade em um variável que representa a quantidade de vezes que aquela face foi obtida no lançamento do dado. Como temos 6 possibilidades de faces, devem ser criadas 6 variáveis para armazenar o total de vezes que cada face foi obtida. Vamos fazer simulação para a seguinte seqüência de lançamentos do dados:

Número do Lançamento	1	2	3	4	5	
Face obtida no lançamento	3	5	1	5	4	0

Suponha que antes de fazer a primeira leitura, a soma dos valores obtidos até o momento é zero, e também contamos zero vezes o número que cada face do dado foi obtida. Na primeira entrada (referente ao valor obtido no primeiro lançamento do dado) obtemos o valor 3, que é então adicionado ao valor anterior da soma total, que é zero, obtendo o valor 3. No contador de número de vezes que a face 3 apareceu devemos somar 1, obtendo 1. O valor obtido na segunda entrada é 5, e nesse caso, a soma total passa a ser $5 + 3 = 8$, e devemos somar 1 no número de vezes que a face 5 apareceu, totalizando 1. Ao fazer a terceira entrada, obtemos o valor 1 e a soma total passa a ser $8 + 1 = 9$ e devemos somar 1 no contador de vezes que a face 1 saiu, totalizando 1. Na quarta leitura, obtemos o valor 5, que adicionado ao valor atual da soma é $9 + 5 = 14$, e como já obtivemos o valor 5 anteriormente, ao incrementarmos o contador para essa face, chegamos ao valor de 2. O próximo valor obtido é 4, que deve ser adcionado à soma geral, que nesse caso fica como 14+4=18 e o contador do número de vezes que a face 4 saiu é incrementada de uma unidade. Na entrada seguinte, obtemos o valor 0 que é exatamente o valor que esperamos como sentinela para finalizar o laço. Então, o laço é finalizado e apresentamos na saída os valores da soma total e o número de vezes que cada face do dado foi obtido. Nesse caso, temos como saída:
Soma total obtida: 18
Num. de vezes que a face 1 foi obtida: 1
Num. de vezes que a face 2 foi obtida: 2
Num. de vezes que a face 3 foi obtida: 3
Num. de vezes que a face 4 foi obtida: 4
Num. de vezes que a face 5 foi obtida: 5
Num. de vezes que a face 6 foi obtida: 6

Esse comportamento apresentado pode ser obtido com o uso do padrão elementar de Repetição com Sentinela, que deve executar o cálculo para a geração da soma total e também o incremente do valor de cada variável que representa o número de vezes que face foi obtida. Os cálculos citados devem

ser repetivos até que seja obtido o valor 0 como entrada, que é o sentinela esperado nesse programa.

Solução para o Problema do dados .

```
01  public static void main(String[] args) {
02    int soma;  /* armazena a soma total dos lancamentos */
03    int lcto;  /* ultimo valor de lancamento lido */
04    int f1;    /* numero de vezes que foi obtida a face 1 num lancamento */
05    int f2;    /* numero de vezes que foi obtida a face 2 num lancamento */
06    int f3;    /* numero de vezes que foi obtida a face 3 num lancamento */
07    int f4;    /* numero de vezes que foi obtida a face 4 num lancamento */
08    int f5;    /* numero de vezes que foi obtida a face 5 num lancamento */
09    int f6;    /* numero de vezes que foi obtida a face 6 num lancamento */

10    soma = 0;
11    f1 = 0;
12    f2 = 0;
13    f3 = 0;
14    f4 = 0;
15    f5 = 0;
16    f6 = 0;

17    lcto = readInt();

18    while (lcto != 0) {
19      soma = soma + lcto;

20      if (lcto == 1)
21        f1 = f1 + 1;
22      else if (lcto == 2)
23        f2 = f2 + 1;
24      else if (lcto == 3)
25        f3 = f3 + 1;
26      else if (lcto == 4)
27        f4 = f4 + 1;
28      else if (lcto == 5)
29        f5 = f5 + 1;
30      else if (lcto == 6)
31        f6 = f6 + 1;

32      lcto = readInt();
33    }

34    writeInt("Soma total obtida:", soma);
35    writeInt("Num. de vezes que a face 1 foi obtida: ", f1);
36    writeInt("Num. de vezes que a face 2 foi obtida: ", f2);
37    writeInt("Num. de vezes que a face 3 foi obtida: ", f3);
38    writeInt("Num. de vezes que a face 4 foi obtida: ", f4);
39    writeInt("Num. de vezes que a face 5 foi obtida: ", f5);
40    writeInt("Num. de vezes que a face 6 foi obtida: ", f6);
```

```
41 }
```

No programa solução apresentado, note que foi utilizada o padrão elementar de Seleção Sequencial (entre as linhas 20 e 31) para escolher a variável que será incrementada, representando a face obtida no lançamento do dado.

4 Plugin de Depuração de Programas para o Dr. Java

4.1 descrição da tela do sistema de depuração

descrever o visualizador do programa, visualizao do caso de teste usado para fazer a depuração, as hipóteses de falha e as predições do aluno.

4.2 exemplo de uma sessão de depuração