Lendo um arquivo em Java

```
import java.io.*;
…
try {
        BufferedReader br = new BufferedReader(new FileReader("/tmp/hello.txt"));
        String line = null;
        do {
                line = br.readLine();
                if (line != null)  System.out.println(line);
        } while(line != null);
        br.close();
} catch (IOException e) {
          e.printStackTrace();
}
```

Universidade de São Paulo
Brasil

Aula 10 – Persistência em Arquivos

Armazenando dados para processamento posterior

MAC0321 - Laboratório de Programação Orientada a Objetos

Professor: Marcelo Finger **(mfinger@ime.usp.br)**

Departamento de Ciência da Computação
Instituto de Matemática e Estatística

Tópicos

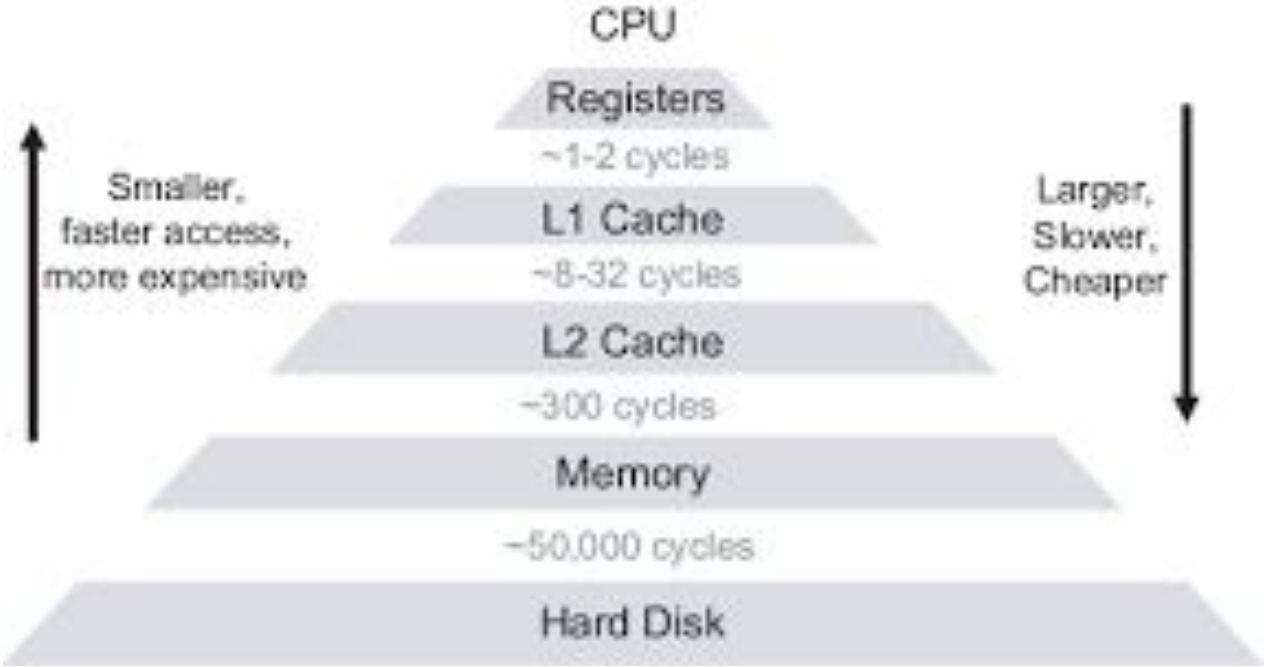
1. Persistência de objetos;
2. Padrão Data Access Object (DAO);
3. DAO usando arquivo.
4. JDBC básico;
5. Biblioteca de acesso a arquivo do Java
6. Leitura de Teclado

Persistência de Objetos em Java

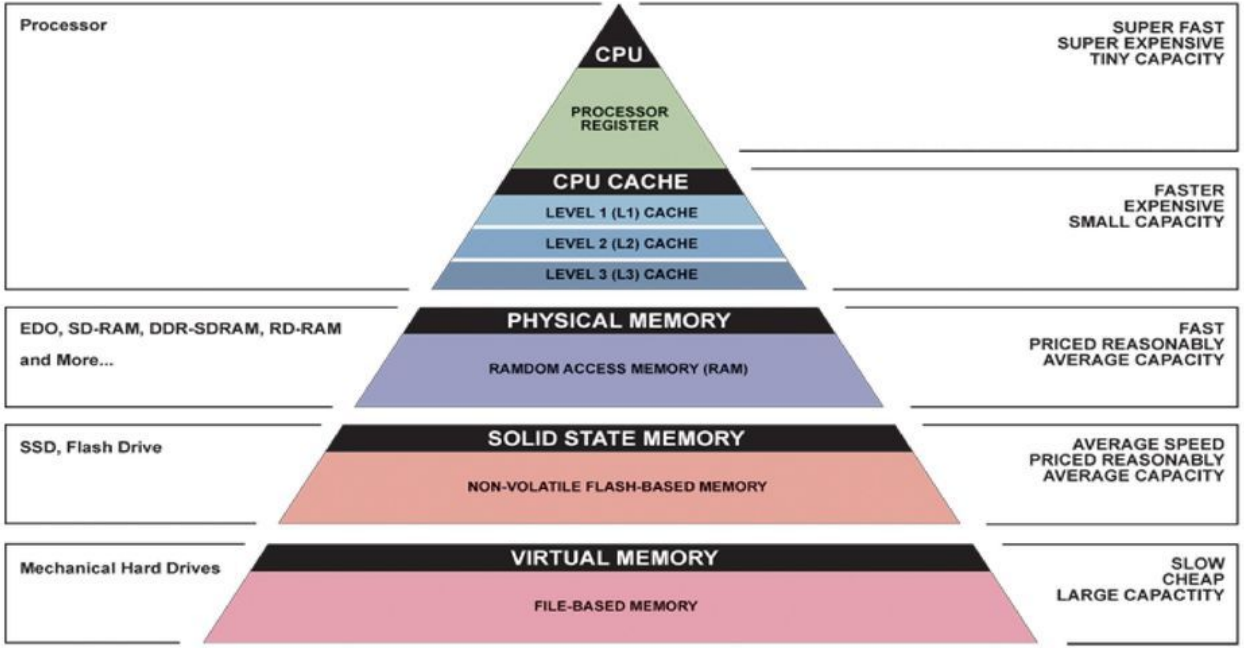
Persistência de objetos

- Como armazenar dados para serem processados depois?
- Temos a hierarquia de memória:
  - Registradores;
  - Memória cache;
  - Memória RAM;
  - Disco rígido;
  - CD-RW/DVD-RW/Fita...
- Somente do disco rígido para baixo o armazenamento de dados é persistente!

Hierarquia de Latência



Hierarquia Moderna



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Criando um arquivo em Java

```
import java.io.PrintWriter;

...

try {
      PrintWriter writer = new PrintWriter("hello.txt", "UTF-8");
      writer.println("Hello World!");
      writer.println("Meu primeiro arquivo em Java!");
      writer.close();
   }
catch(FileNotFoundException | UnsupportedEncodingException e) {
      System.out.println(e);
}
```

Criando um arquivo em Java

- Na primeira linha instanciamos um objeto da classe PrintWriter, passando como parâmetros para o construtor o nome e a **codificação do arquivo**.
  - Isso é o que pode acontecer quando você não passa a codificação correta.
- Temos um objeto da classe PrintWriter, só precisamos usar o método correto para escrever no arquivo aberto.
  - Repare o nome do método. Familiar?

Onde armazenar os dados?

- Vimos uma maneira de armazenar os dados (texto puro), mas existem outras maneiras.
- Outras opções:
  - Banco de dados;
    - Relacionais: SQL;
    - Não-relacionais: NoSQL.
  - Arquivos binários.
- A melhor opção depende da aplicação!

E a Orientação a Objeto, como fica?

- Podemos simplesmente instanciar um objeto da classe PrintWriter cada vez que precisarmos escrever num arquivo, usar e fechar o objeto depois de usarmos;
- Mas e se precisarmos modificar o código?
  - A aplicação cresceu e agora um arquivo texto puro não dá mais conta. Precisamos usar um banco de dados relacional!
  - Ctrl+F no código inteiro para ir substuindo cada entrada do PrintWriter...

Padrão DAO

- Ao invés disso, podemos usar um padrão de projeto chamado **Data Access Object (DAO)**;
  - Nele, criamos uma classe que dá acesso as operações permitidas (inserir, deletar, modificar, etc.);
  - Se precisarmos alterar o backend, faremos só na classe criada.


GREAT SUCCESS!
evilmilk.com

Padrão Data Access Object (DAO)

Padrão DAO

- O padrão DAO é um padrão estrutural
- Permite isolar a camada de aplicativo/ negócios da camada de persistência
- Persistência: geralmente um banco de dados relacional, mas pode ser qualquer outro mecanismo
- Usa uma API abstrata.

Padrão DAO: Exemplo

```
public interface StudentDAO {
        public boolean insertStudent(Student student);
        public Student getStudent(Student student);
        public List<Student> getAllStudents();
        public boolean updateStudent(Student student);
        public boolean deleteStudent(Student student);
}
```

Acessando banco de dados em Java

- A maioria dos projetos de software hoje utilizam:
  - Uma linguagem de OO, como Java;
  - Um banco de dados relacional, como MySQL.
- Os dois modelos são bem diferentes entre si!
- Como resolver?

Alternativa 1: Força bruta!

```
conn = ConnectionFactory.createConnection();
String sql = "SELECT nome, sobrenome FROM funcionario";
ps = conn.prepareStatement(sql);
ResultSet rs = ps.executeQuery();
while(rs.next()){
    String nome = rs.getString("nome");
    String sobreNome = rs.getString("sobrenome");
    String nomeCompleto = nome + sobreNome;
}
```

Mais coisas...

```
conn = ConnectionFactory.createConnection();

String sql = "SELECT nome, sobrenome, idade FROM funcionario";

ps = conn.prepareStatement(sql);

ResultSet rs = ps.executeQuery();

while(rs.next()){
        String nome = rs.getString("nome");
        String sobreNome = rs.getString("sobrenome");
        int idade = rs.getInt("idade");
        String nomeCompleto = nome + sobreNome;
}
```

E mais coisas...

```
conn = ConnectionFactory.createConnection();

String sql = "SELECT codigo, nome, sobrenome, idade, salario FROM funcionario";

ps = conn.prepareStatement(sql);

ResultSet rs = ps.executeQuery();

while(rs.next()){

                    int codigo = rs.getInt("codigo");

                    String nome = rs.getString("nome");

                    String sobreNome = rs.getString("sobrenome");

                    int idade = rs.getInt("idade");

                    String nomeCompleto = nome + sobreNome;

                    Double salario = rs.getDouble("salario");

                    System.out.printf("Código %d: %s - %d | Salário: %f \n",codigo, nomeCompleto, idade, salario);

}
```

DAO Usando JDBC

Outra maneira mais elegante

```
public class StudentJDBC implements StudentDAO {

@Override
public boolean insertStudent(Student student) {
        db = conectar();
        db.executeUpdate("INSERT INTO Pessoa VALUES('" +
                                student.getRg() + "', '" +
                                student.getNome() + "'," + student.getIdade() +  ",'" +
                                student.getCidade() + "','" + student.getEstado() + "')");
        return true;
}
...
```

Bibliotecas de acesso a disco

- Vimos o PrintWriter (arquivos texto puro) e o JDBC (banco de dados relacionais);
- Há outras bibliotecas, como
  - **java.io** (baixo nível)
  - **java.nio** (non-blocking IO, para programas de alto desempenho)

  para trabalhar com arquivos binários, etc.
- Cada uma tem sua aplicação!

Leitura do Teclado

Leitura do Teclado: Scanner

```
import java.util.Scanner;
Scanner keyboard = new Scanner(System.in);
System.out.println("enter an integer: ");
int myint = keyboard.nextInt();
keyboard.nextLine();  // Consume newline left-over
System.out.println("enter a msg: ");
String msg = keyboard.nextLine();
```

The `nextInt()` Problem

- nextInt() não lê o "\n" seguinte
- Então, depois de um nextInt(), se você for ler um texto, você deve primeiro consumir esse "\n" (nova linha)
- Faça isso usando nextLine()
- Se você for ler outro número inteiro, isso não é problema, pois o próximo nextInt() consumirá a nova linha.

Avoiding the `nextInt()` Problem

```
Scanner input = new Scanner(System.in);
System.out.println("enter an integer: ");
    int myint = 0;
    try {
        myint = Integer.parseInt(input.nextLine());
    } catch (NumberFormatException e) {
        e.printStackTrace();
    }
    System.out.println("enter a msg: ");
    String msg = input.nextLine();
```

Agora descubra sozinho ...

- … como pegar o próximo double.

Double

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
float PI = 3.1416f;
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Radius?");
 try{
     String raio = br.readLine();
     int r = Double.parseDouble(raio);
     System.out.println("Circle area is: " + PI*r*r + " Perimeter: " +PI*2*r);
 }
 catch(Exception e){
      e.printStackTrace();
 }
```

Lista de exercícios

No computador com o Eclipse

Entrega até o final do dia

MAC321
Lab POO

- Professor: Marcelo Finger

  E-mail: mfinger@ime.usp.br