Código 1 - Revisão dos conceitos da linguagem de programação Java

Resumo: O código apresentado exemplifica o uso de conceitos fundamentais de Java, como encapsulamento, orientação a objetos, manipulação de coleções e leitura de dados do usuário. Ele demonstra a criação de objetos Pessoa, a captura de dados via Scanner, e o armazenamento desses objetos em uma lista ArrayList, para exibição posterior. As principais referências utilizadas para elaborar este material incluem o livro "Java: Como Programar" (Deitel) para os conceitos de orientação a objetos e encapsulamento, além da documentação oficial da linguagem Java para detalhes técnicos específicos.

Será detalhado abaixo cada parte das classes Principal e Pessoa, explicando os conceitos usados nos códigos.

Classe “Pessoa”

```
public class Pessoa{
    private String nome;
    private int idade;
```

Declaração da classe Pessoa com dois atributos privados: nome e idade. Em Java, é uma boa prática encapsular os dados, mantendo os atributos privados e fornecendo métodos públicos para acessá-los.

```
public Pessoa(){
    this("", 0);
}
```

Construtor padrão que chama outro construtor usando this, passando valores padrão ("" para nome e 0 para idade).

```
public Pessoa(String nome, int idade){
    setNome(nome);
    setIdade(idade);
}
```

Construtor que aceita parâmetros para inicializar os atributos nome e idade. Ele chama os métodos setNome e setIdade para atribuir os valores aos atributos.

```
public void setNome(String nome){
    this.nome = nome;
}

public String getNome(){
    return this.nome;
}
```

Métodos setNome e getNome que permitem definir e acessar o valor do atributo nome. O this é utilizado para referenciar o atributo da instância atual.

```
public void setIdade(int idade){
    this.idade = idade;
}
```

```
public int getIdade(){
    return this.idade;
}
```

Métodos setIdade e getIdade para definir e acessar o valor do atributo idade.

```
@Override
public String toString(){
    return String.format("Pessoa{ Nome: %s, Idade: %d}", getNome(), getIdade());
}
```

O método toString é sobrescrito para fornecer uma representação em string da instância Pessoa. Aqui, o String.format é utilizado para formatar a string de saída, mostrando os valores de nome e idade.

Classe "Principal"

```
import java.util.Scanner;
import java.util.ArrayList;
```

Essas linhas de código importam as classes Scanner e ArrayList das bibliotecas Java. O Scanner é utilizado para entrada de dados, enquanto o ArrayList é uma coleção que permite armazenar objetos de forma dinâmica.

```
public class Principal{
```

Define a classe Principal. Em Java, toda aplicação começa com a definição de uma classe. O nome da classe deve ser o mesmo do arquivo .java correspondente.

```
public static void main(String[] args){
```

Este é o método principal (main), onde a execução do programa começa. É um método estático, o que significa que pode ser chamado sem a necessidade de criar uma instância da classe Principal.

```
Scanner input = new Scanner(System.in);
```

Cria um objeto Scanner para ler a entrada do usuário a partir do console. O System.in especifica que a entrada será lida do console.

```
Pessoa objPessoa = new Pessoa();
```

Cria um novo objeto da classe Pessoa utilizando o construtor padrão (sem parâmetros).

```
System.out.println("Digite os dados da pessoa.");
System.out.print("Nome: ");
String nomePessoa = input.nextLine();
```

Essas linhas exibem uma mensagem solicitando o nome da pessoa e, em seguida, leem a entrada do usuário.

```
System.out.print("Idade: ");
int idade = input.nextInt();
```

Solicita a idade da pessoa e lê o valor como um inteiro.

```
objPessoa.setNome(nomePessoa);
objPessoa.setIdade(idade);
```

Os métodos setNome e setIdade são chamados para atribuir os valores lidos à instância objPessoa.

```
ArrayList<Pessoa> listaPessoas = new ArrayList<Pessoa>();
```

Cria uma ArrayList que armazenará objetos do tipo Pessoa. O ArrayList é uma coleção dinâmica que pode crescer conforme novos elementos são adicionados.

```
listaPessoas.add(new Pessoa("José", 30));
listaPessoas.add(new Pessoa("Mari", 25));
listaPessoas.add(new Pessoa("Miguel", 40));
listaPessoas.add(objPessoa);
```

Adiciona novos objetos Pessoa à lista, incluindo o objeto objPessoa que foi criado anteriormente.

```
for(Pessoa auxPessoa : listaPessoas){
    System.out.println(auxPessoa.toString());
}
```

Este laço for-each percorre cada objeto Pessoa na lista e exibe as informações usando o método toString.