Universidade de São Paulo
Brasil

Aula 09 – Bibliotecas de Java

Reutilização de boas implementações no desenvolvimento de software

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. Um pouco do que há disponível em Java
2. Bibliotecas
3. Estruturas de dados em Java
4. <Generics>, e como usá-los
5. Iteradores e seus primos
6. ChatGPT

O que já existe para Java

Onde achar as bibliotecas

- Elas tem documentação on-line em:
  - http://docs.oracle.com/javase/17/docs/api/index.html
  - Organização dos diversos pacotes
  - Para cada pacote temos:
    - Interfaces
    - Classes
    - Enums
      - Lista de constantes pré-definidas
    - Exceptions

Exemplo: java.math

- Classes
  - BigDecimal
  - BigInteger
  - MathContext
- Enums
  - RoundingMode

Bibliotecas Java

Olhando mais de perto

- Class BigInteger
  - java.lang.Object
  - java.lang.Number
  - java.math.BigInteger
- All Implemented Interfaces:
  - Serializable
  - Comparable<BigInteger>

Serializable

- public interface Serializable
  - A serialização de uma classe é habilitada pela classe que implementa o java.io.
  - Interface serializável. As classes que não implementam esta interface não terão nenhum de seus estados serializado ou desserializado.
  - Todos os subtipos de uma classe serializável são serializáveis.
  - A interface de serialização não possui métodos ou campos e serve apenas para identificar a semântica de ser serializável.

Interface Comparable<T>

- T
  - o tipo de objetos aos quais este objeto pode ser comparado
- Um único método :)
- compareTo
  - int compareTo(T obj)

<Generics> em Java

Visão geral de generics (<T>)

- Genéricos permitem que os tipos (classes e interfaces) sejam parâmetros ao definir classes, interfaces e métodos.
- Muito parecido com os parâmetros formais mais familiares usados em declarações de método, os parâmetros de tipo fornecem uma maneira de reutilizar o mesmo código com entradas diferentes.
- A diferença é que as entradas para parâmetros formais são valores, enquanto as entradas para digitar parâmetros são tipos.
- *Origem: templates de C++, mas com semântica mais bem definida*

Exemplo de uso

Classe Box: uma caixinha contendo um objeto qualquer, definido em tempo de execução

```
public class Box {
    private Object object;
    public void set(Object object) {
      this.object = object;
    }
    public Object get() {
      return object;
    }
}
```

Exemplo de uso com generics

- Classe Box: infinitas caixas, uma para cada tipo T

```
public class Box <T> {
    private T object;

    public void set(T object) {
      this.object = object;
    }
    public T get() {
      return object;
    }
}
```

- Usando a Box: Box<String> stringBox;

Vários generics

```
public interface Pair<K, V> {
    public K getKey();
    public V getValue();
}

public class OrderedPair<K, V> implements Pair<K, V> {
    private K key;
    private V value;
    public OrderedPair(K key, V value) {
                this.key = key;
                this.value = value;
    }
  public K getKey()   { return key; }
  public V getValue() { return value; }
}
```

Os Tipos de um Generics

- Tem de ser uma classe.
- Não pode ser um tipo primitivo (int, double, etc)
- Se temos um generics de Integer
  - Podemos passar um int
- Isso vale para os outros tipos
- Generics funciona muito bem para estruturas de dados

ArrayList

java.lang.Object

java.util.AbstractCollection<E>

java.util.AbstractList<E>

java.util.ArrayList<E>

- All Implemented Interfaces:
  - Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess
- Direct Known Subclasses:
  - AttributeList, RoleList, RoleUnresolvedList

Alguns Métodos do ArrayList

boolean add(E e)

Appends the specified element to the end of this list.

void add(int index, E element)

Inserts the specified element at the specified position in this list.

boolean addAll(Collection<?extends E> c)

Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.

boolean addAll(int index, Collection<? extends E> c)

Inserts all of the elements in the specified collection into this list, starting at the specified position.

Métodos do ArrayList (cont)

void clear()

Removes all of the elements from this list.

Object clone()

Returns a shallow copy of this ArrayList instance.

boolean contains(Object o)

Returns true if this list contains the specified element.

E get(int index)

Returns the element at the specified position in this list.

Vantagens dos Generics:

Type-Safety/Segurança de Tipos

```
List list = new ArrayList();

list.add(10);

list.add("10");
```

Com Generics é necessário especificar o tipo de objeto que precisamos armazenar.

```
List<Integer> list = new ArrayList<Integer>();

list.add(10);

list.add("10"); // pau de compilação
```

Vantagens dos Generics: cast desnecessário

```
List list = new ArrayList();

String s = (String) list.get(0); //cast
```

Com Generics não é necessário fazer o cast

```
List<String> list = new ArrayList<String>();

list.add("Olá");

String s = list.get(0);
```

Vantagens dos Generics: Verifica Tipos na Compilação

```
List<String> list = new ArrayList<String>();

list.add("hello");

list.add(32); //Compile Time Error
```

Uso de numerator (*enumerador*)

- Forma genérica de se percorrer diversas estruturas
- Interface simples com dois métodos
  - boolean hasMoreElements()
  - E next()
- Onde E é um generics
- Vantagem pode se mudar a implementação sem mudar a interface
- Ver Padrão: Iterator

Outra interface interessante: Iterator

default void forEachRemaining(Consumer<?super E> action)

Performs the given action for each remaining element until all elements have been processed or the action throws an exception.

boolean hasNext()

Returns true if the iteration has more elements.

E next()

Returns the next element in the iteration.

default void remove()

Removes from the underlying collection the last element returned by this iterator (optional operation).

Exemplo de uso

http://www.javapractices.com/topic/TopicAction.do?Id=125

https://blog.udemy.com/java-iterator/

ChatGPT

Lista de exercícios

No computador com o Eclipse

Entrega até o final do dia

Usando o chatGPT

- Crie uma seção no ChatGPT
- Diga que você vai querer gerar código em Java
- Inserir uma adaptação do exercício 1 da Lista

  Usando a biblioteca Swing do Java, crie uma interface simples para o programa de Fatorial usando BigDecimal. Basta colocar um lugar onde o usuário possa entrar um número, um botão para iniciar o cálculo e um lugar para mostrar o resultado. Separe o código da interface (a View) do código que faz o cálculo de Fatorial (o Controller). Não esqueça do tratamento de exceções, o programa não deve retornar resultados errados nem fazer o usuário se deparar com uma mensagem de erro esquisita!

- Ver a resposta, tentar rodar e entender o que ele sugeriu!
- Daqui duas aulas: refatorar o código antes de alterá-lo para apresentar números grandes.

MAC321
Lab POO

- Professor: Marcelo Finger

  E-mail: mfinger@ime.usp.br