Tipos Genéricos em Java

Computação Orientada a Objetos

Prof. Flávio Luiz Coutinho

EACH/USP

Tipos Genéricos

- Motivação: classe **Pilha**.
  - Implementa estrutura de dados **pilha**.
  - Coleção de elementos.
  - Especifica como os elementos entram e saem da coleção.
  - Último a entrar é o primeiro a sair (LIFO – last in, first out).
  - Métodos **push** e **pop**.
- Código.

Motivação

- É necessário uma implementação própria da classe **Pilha** para cada tipo de elementos com os quais queremos trabalhar.
- Solução: classe “genérica” (ainda sem usar recursos de tipos genéricos de fato).

- Código (“forçando” a versão 1.4).

Tipos Genéricos

- Problemas da nossa versão genérica:
  - Conversões (**cast**) de tipo devem ser explícitas.
  - Possíveis erros de execução devido a pilha conter um objeto de uma classe não esperada (**ClassCastException**).
  - **Boxing/unboxing** de tipos primitivos (boolean, byte, char, short, int, long, float e double): uso de classes empacotadoras.

Tipos Genéricos

- A partir do Java 1.5 foram disponibilizados recursos para a implementação de classes genéricas:
  - Tipos genéricos.
  - Auto boxing/unboxing.

- Código ilustrando auto boxing/unboxing.

Tipos Genéricos

- Outro exemplo: método **printArray**.
  - Imprime o conteúdo de um array.
  - Método sobrecarregado para realizar uma operação semelhante em diferentes tipos de dados.
  - Versões para **Integer**, **Double** e **Character**.

- Código.

Tipos Genéricos

- Tipos **genéricos**:
  - Permite a declaração de métodos e classes genéricas.
  - Uma única implementação é capaz de lidar com diversos tipos de classes.
  - Recurso poderoso para promover a reutilização de código.

Tipos Genéricos

- Métodos genéricos: **printArray**.
- Classes genéricas: **PilhaGenerica**.
- Benefícios do uso de tipos genéricos:
  - Reutilização de código.
  - Verificação em tempo de compilação (evita exceções do tipo **ClassCastException**).
  - Não há a necessidade de conversões de tipos (**cast**): código mais limpo e enxuto.

Tipos Genéricos

- Formalização:
  - **Seção de parâmetros de tipo**: <T, E, …, X>.
  - Contém um ou mais **parâmetros de tipo**, ou **variável de tipo**.
  - Parâmetros de tipo podem ser usados para especificar:
    - Tipo de retorno.
    - Tipos de parâmetros.
    - Tipos de variáveis locais.
  - **Parâmetros de tipo**: marcadores de lugar para os **argumentos de tipos reais** (tipos dos argumentos passados ao método ou classe genéricos).

Tipos Genéricos

- Paralelo entre **métodos** e **métodos genéricos**:
  - Ambos são recursos visando reutilização de código.
- Métodos:
  - Mesmas operações sobre dados variáveis, mas de tipo comum.
- Métodos genéricos:
  - Mesmas operações sobre tipos e dados variáveis.
  - Tipos dos dados são parametrizados.
  - Maior flexibilidade.

Tipos Genéricos

- O que acontece durante a compilação?
  - Código .java → bytecodes (.class)
  - Remoção da seção de parâmetros de tipo.
  - Substituição dos parâmetros de tipo por tipos reais.
  - Remoção + substituição: **erasure**.
  - Por padrão, tipos genéricos → Object.
  - Versões compiladas não são diferentes das versões pseudo-genéricas (usando Object).
  - *Syntatic sugar*: recursos de sintaxe que facilitam a produção/leitura de código.

Tipos Genéricos

- Outros exemplo de *syntatic sugar*:
  - for (Element e : array) { … }
  - Auto-boxing/unboxing

- Exemplo **printArray**: poderia ser implementado usando Object ao invés do tipo genérico.
- Benefício é mais aparente quando se usa parâmetros de tipo para especificar o tipo de retorno do método (como no exemplo **PilhaGenerica**).

Tipos Genéricos

- Exemplo **maximum**.

Tipos Genéricos

- Muitas classes da biblioteca do Java são implementadas como classes genéricas, como iremos estudar em Collections.
- Há também algumas interfaces genéricas como a **Comparable<T>**, implementada por todas as classes empacotadoras e outras como a classe String.
- Exemplo **maximum** usando a interface genérica **Comparable<T>**.
  - “Limite superior” de um parâmetro de tipo.
  - Durante o **erasure**, o tipo genérico é substituído pelo “limite supeior”.

Tipos Genéricos

- Outros exemplo: método genéricos para testar **Pilha<T>**.

Tipos Genéricos

- Tipos brutos:
  - Object implícito usado como tipo de parâmetro.
  - Importância para retrocompatibilidade de código.
- Coleções Java:
  - Antes: guardavam referências para Objects.
  - Agora: guardam referências para tipos parametrizados.
  - O curioso é que depois do processo de **erasure** não há diferença!