UNIDADE 3 – GENÉRICOS

1.[MB - 2010 - MM] Quanto ao uso de Generics na linguagem de programação Java 1.5 ou superiores, é correto afirmar que:

(a) Os métodos genéricos permitem que programadores especifiquem, com várias declarações semelhantes de métodos, um conjunto de métodos polimórficos relacionados, onde são diferentes os parâmetros esperados por cada implementação de métodos dentro da mesma classe.

(b) Todas as declarações de métodos genéricos têm uma seção de parâmetro de tipo delimitado por colchetes angulares (< e >) que precede o tipo de retorno do método. Além disso, os nomes de parâmetros de tipo precisam ser únicos entre os diferentes métodos genéricos da mesma classe.

(c) Quando o compilador traduz um método genérico em bytecode Java, ele remove a seção de parâmetro de tipo e substitui por tipos reais. Esse processo é conhecido como Erasure.

(d) A tecnologia de Generics permite escrever classes genéricas, onde todos os métodos contidos nesta classe são genéricos, exigência esta que é avaliada em tempo de execução para bytecode Java.

(e) Uma classe não-genérica pode ser derivada de uma classe genérica. Isto se dá devido a especificação de Polimorfismo do modelo Orientado a Objetos.

2.[VUNESP - 2011 - Prefeitura de Diadema] Considere o seguinte trecho de código na linguagem de programação Java:

```
public void imprimirCollection(Collection c) {
   Iterator i = c.iterator();
   for(int k = 0;k < c.size();k++)
{ System.out.println(i.next()); } }
```

O código que permite a impressão (chamada do método toString()) de cada objeto da Collection c, utilizando o recurso Generics da linguagem Java, é

(a) void imprimirCollection(Collection c) { for(Object o:c) { System.out.println(o); } }
(b) void imprimirCollection(Collection c) { for(Object o:c) { System.out.println(o); } }
(c) void imprimirCollection(Collection c) { for(int i = 0; i < c.size(); i++) { System.out.println(c.at(i)); } }
(d) void imprimirCollection(Collection c) { for(int i = 0; i < c.length; i++) { System.out.println(c[i]); } }
(e) void imprimirCollection(Collection < ? > c) { for(Object o:c) { System.out.println(o); } }

3.[PUC-PR - 2010 - COPEL] Sobre Java e a extensão chamada Generics adicionada a partir da versão 5 da linguagem Java, analise a classe abaixo e assinale a alternativa CORRETA, caso algum programa precise declarar uma variável do tipo Mapa:

```
1: public class Mapa<K,V> {
2:     ...
3:     V inserir(K chave, V valor) {
4:         ...
5:     }
6: }
```

(a) Não é obrigatório informar um tipo para K e V.

(b) Para criar uma variável do tipo Mapa, é obrigatório informar um tipo para K e V.

(c) Para criar uma variável do tipo Mapa, é obrigatório informar um tipo para K ou V.

(d) Não é obrigatório informar um tipo para K e V, porém, ao instanciar um objeto do tipo mapa, um erro de tempo de execução será gerado.

(e) A linha 1 está incorreta, pois não é permitido passar mais de um tipo formal de parâmetro.

4. [CESGRANRIO - 2011 - Petrobrás] Coleções consistem em objetos que permitem manter diversos elementos armazenados como uma unidade. Elas incluem as implementações de várias estruturas de dados, sendo um importante fator de ganho em eficácia e eficiência para o desenvolvedor que souber usá-las.

Nesse sentido, as coleções em Java

a) incluem a classe Vector, que é a representação mais veloz de vetores, mas é limitada pela falta de sincronização e pela limitação do número de elementos armazenados.

b) incluem a classe BTreeMap, que é uma representação direta da estrutura de árvore B e que é a única classe das coleções cujo construtor necessita de um parâmetro (a ordem da árvore).

c) incluem a interface Map, que permite um mapeamento chave-valor que pode ser implementado de maneiras diferentes, como através de hashes e árvores.

d) podem ser definidas para implementações com genéricos, que são restritos às classes da GenericCollection, uma extensão de Collections criada especificamente para tal propósito.

e) contêm métodos especiais chamados isTree(), isMap() e isSet(), entre outros, para que o usuário possa saber que tipo de estrutura de dados está sendo efetivamente implementada.

5. [CIAAR - 2009 - CIAAR] Informe se é falso (F) ou verdadeiro (V) o que se afirma abaixo sobre estruturas de dados em linguagem Java. A seguir, indique a opção com a sequência correta.

( ) Uma coleção é uma estrutura de dados capaz de armazenar objetos. Ela pode crescer e encolher dinamicamente.
( ) Collection é uma interface que declara métodos capazes de inserir e remover coleções de objetos.
( ) Uma List é uma interface com métodos para implementar coleções. As classes ArrayList, Vector e LinkedList implementam estes métodos da interface List.
( ) Um Set é uma coleção que não pode conter nenhum elemento duplicado. Entre as classes que implementam esta coleção encontramos HashMap, ArrayList e TreeMap.

a) V – V – V – F. b) V – F – V – F. c) F – V – V – F.

d) V – V – F – V.

6.[FCC - 2009 - TJ/SE] Uma lista Java é uma coleção ordenada de elementos do mesmo tipo, conhecida por sequência. Os elementos de uma lista podem ser acessados pela sua posição, isto é, seu índice e são derivados da interface

(a) java.util.LinkedList, que estende a interface Collection
(b) java.util.Collection, que estende a interface Set
(c) java.util.Set, que estende a interface Collection
(d) java.util.Collection, que estende a interface List
(e) java.util.List, que estende a interface Collection

7.[CESPE - STJ - 2004] O Java collections framework da API Java JSE possui um conjunto de interfaces e implementações que define estruturas usadas para manipular coleções de objetos. Acerca das interfaces e implementações das estruturas do Java collections framework, julgue os itens que se seguem. As interfaces fundamentais do framework estão associadas à identificação de funcionalidades típicas de estruturas de dados clássicas. Assim, a interface java.util.List está ligada a estruturas de listas, a interface java.util.Set está associada a estruturas do tipo conjuntos e a interface java.util.Map refere-se a estruturas do tipo mapas. Set, List e Map possuem a interface abstrata java.util.Collection como superinterface?

(a)Certo (b)Errado

8. Quais das seguintes afirmativas são verdadeiras sobre o código abaixo?

```
class Generics17{
        public static void main(String[ ] args){
            List<Object> list=new LinkedList<Object>();
            list.add("A");
            addToList("B", list);
            System.out.println(list);
        }
        static void addToList(Object o, List<?> l){
            l.add(o);
        }
}
```

(a) O código irá compilar se List< ? > for substituído por List
(b) O código irá compilar se List for substituído por List

(c) O código irá compilar se List for substituído por LinkedList
(d) O código compila normalmente

9. Qual é o resultado da compilação e execução do código abaixo?

```
List<? extends Object> aList = new Vector<String>();
aList.add("USA");
aList.add("Russia");
aList.add("UK");
for(String s : aList)
        System.out.print(s);
```

(a) Erro de compilação: Sintaxe incorreta usada para declaração do tipo generic aList
(b) Erro de compilação: Objeto Vector não pode ser associado com uma variável List
(c) Compila corretamente e imprime "USAUKRussia"
(d) Compila corretamente e imprime "USA", "UK" e "Russia" em uma ordem que não pode ser prevista
(e) Nenhuma das anteriores

10. [CESPE - STJ – 2004] A ordenação de objetos de um mesmo tipo em uma coleção é feita pela implementação da interface Comparable, implementada em todas as classes que implementam interfaces do collections framework de JSE.

(a) Certa (b) Errada

GABARITO

1 – A; 2 – E; 3 – A; 4 – C; 5 – A; 6 – E; 7 - B; 8 – A;
9 – E; 10 – B