




Lecture 3
Java SE – Programming



presentation

Java Programming – Software App Development
Cristian Toma

D.I.C.E/D.E.I.C – Department of Economic Informatics & Cybernetics
www.dice.ase.ro



Cristian Toma – Business Card



Cristian Toma

IT&C Security Master

Dorobantilor Ave., No. 15-17
010572 Bucharest - Romania

<http://ism.ase.ro>

cristian.toma@ie.ase.ro

T +40 21 319 19 00 - 310

F +40 21 319 19 00



Agenda for Lecture 3 – Summary of JSE





Java Generics, Error @ runtime versus @compile-time

Java Generics



1 Summary of Java Generics

What are advantages of Generics in programming?

Moves the errors from “run-time in compile-time”

Is there any macro-expansion as in C / C + + within Generic programming?

NO

Where is the most intensive usage of Generic programming?

Starting with JDK 5.0 and especially in JCF – Java Collection Framework

ATTENTION!!! For advanced Java Generics concepts please read:
“Sub-typing”, “Wild-Cards”, “Type-Erasure” in the web resources

<http://docs.oracle.com/javase/tutorial/java/generics/>

<http://docs.oracle.com/javase/tutorial/extra/generics/index.html>

<http://java.sun.com>

Java Generics Simple Samples – Generics1.java & Generics4.java

1 Summary of Java Generics

Recommendations for parameters naming are:

- * **E - Element** – intensively uses in JCF - Java Collections Framework
- * **K - Key**
- * **N - Number**
- * **T - Type**
- * **V - Value**
- * **S,U,V** etc. - 2nd, 3rd, 4th types

Please in Ubuntu 12 virtual machine check-out lecture 1 and 2 from **/home/stud/javase** directory

Section Conclusion

Fact: Generics in Java

In few **samples** it is simple to remember: Java generics allows the programmer to use general classes in error prone approach and to provide mechanism for moving the error from runtime to compile time.





JCF – Java Collection Framework, List<E>, Map<K,V>

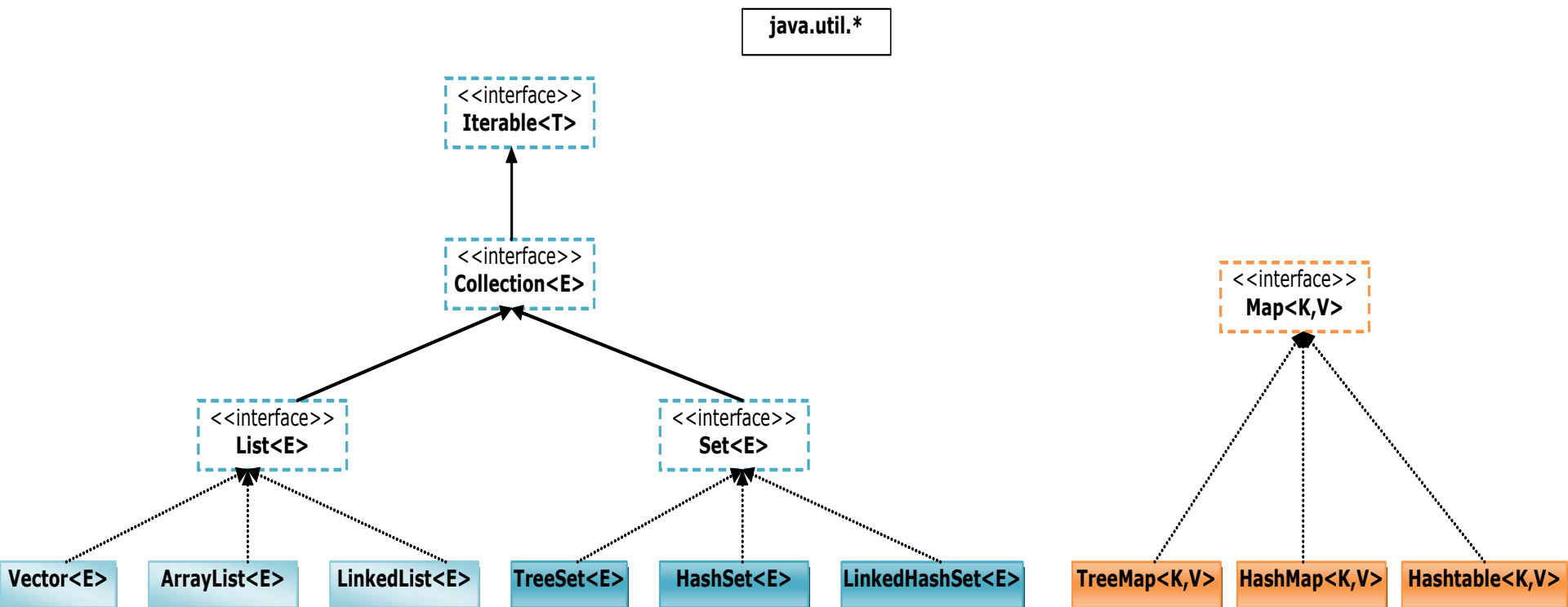
JCF – Java Collection Framework

2. Summary of JCF

JCF – Java Collection Framework Features:

- **JCF** is an hierarchy of classes, abstract classes, interfaces and algorithms, that implements the standard data structures used in programming – vector, list – stack/queue, binary-tree, hash-table
- **JCF** contains interfaces, implementations & algorithms
- **JCF** involves to code with ***interface as type*** style
- The classes hierarchy is based on:
 - ***Collection*** – defines a value for each item in the data structure
 - ***Map*** – defines a pair of items, key-value, for each element/node in the data structure

2. Summary of JCF



2. Summary of JCF

1. In order to go through a data structure from JCF, it is possible to use **foreach** or **iterators** (or partially to use **Enumeration** for classes **Vector** and **Hashtable**)
 - a. `for(Object o : collection) System.out.println(o);`
 - b. `for(Iterator<?> it = collection.iterator(); it.hasNext();)
System.out.println(it.next())`
2. The order of the items/elements into collections/data structures (including for use of sorting algorithms) is given by the implementation of the method “***compareTo(...)***” from the interface **Comparable<T>** or by the implementation of the method “***compare(...)***” from the interface **Comparator<T>**.
3. For optimization and best practice programming, it is recommended for the classes that instantiate objects which are used in hash-data structures, to implement the inherited methods “***hashCode()***” and “***equals(...)***” from class **Object**.

Section Conclusions

JCF – Java Collection Framework is a set of classes, interfaces and algorithms for standard data-structure processing

JCF – presents almost like in C++ STL: containers, iterators, and algorithms

JCF – needs for order of the items in the data-structures to process objects from classes that provide methods for comparing.

In JCF the best practice is to override methods from class Object for equality and hashing value, in order to work with hash data structure

JCF Summary

for easy sharing



Share knowledge, Empowering Minds

Communicate & Exchange Ideas





Questions & Answers!





Thanks!



DAD – Distributed Application Development
End of Lecture 3 – summary of Java SE

