Cours / TP IUT 2012

Introduction to Java.lang*, Java.util.* , Java.io.*

arnaud.nauwynck@gmail.com

Outline

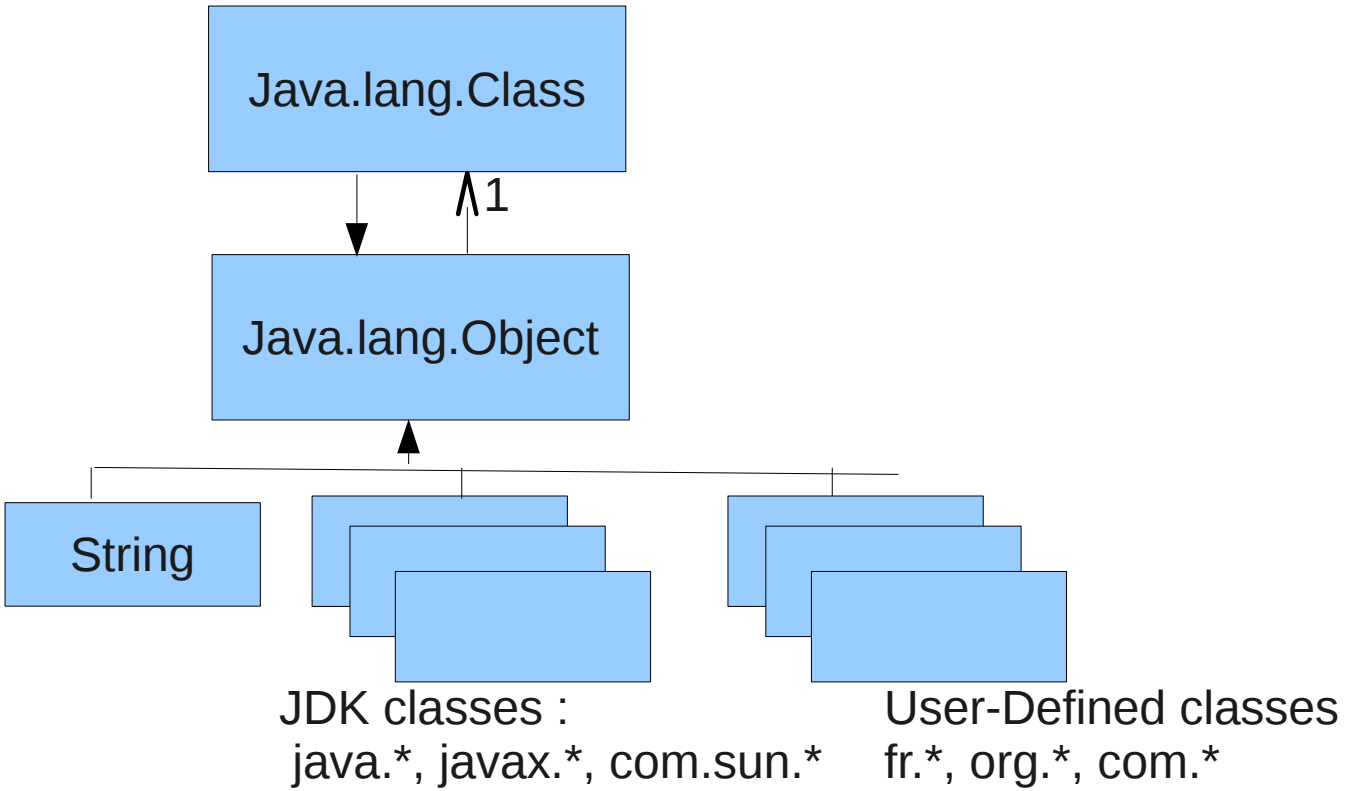
- Java.lang.*
  - Object, Class, String
- Java.util.*
  - List (ArrayList), Object equals()
  - Hash algorithm, HashMap/Set, hashCode()
  - R-B Tree algoritm, TreeMap/Set, compareTo()
- Java.io.*
  - Output/Input Stream
  - PrintStream / StreamReader
  - DataOutput / Input, Object Input/Output Stream

MAY – SHOULD - MUST KNOW

- A java developer MUST KNOW BY HEART 90% of java.lang/util/io.* package...
- Java.lang.String and java.util.ArrayList = the 2 most widely used classes
- Anyway a “person” who doesn't know this is NOT CONSIDERED a Java Developper
- eliminatory questions in jobs recruitement

java.lang.Object

- All Objects are extending “java.lang.Object”
- An Object has 1 (immutable) Class (=Type)
- Java.lang.String are immutable



Class, ClassLoader, Method/Field Introspection

- Class is both
  - **INTERNAL for type-checking**
    ensure integrity and security of the JVM
  - **PUBLIC for introspection / reflection**
- Classes are loaded per **ClassLoader**
  - A Class is loaded at most 1 per classLoader
  - Possibility to have several isolated ClassLoaders
  - => osgi / plugins / application servers architectures
- More in Introspection presentation

Java.lang.String

```
String text = "Hello" + " World";
text += ".";
if (text.charAt(11) == '.') text = text.substring(0, 11);
text = text.replace("a", "b");
byte[] toBytes = text.getBytes();
String text2 = new String(toBytes, Charset.forName("Utf-8"));
if (!text2.equals(text)) throw new IllegalStateException();
```

- String are immutable
- Concatenate '+', Replace … => new String
- Dot not compare with “==” … use “.equals()” !!!!

StringBuilder

- Use StringBuilder for writable/temporary buffer
- Basically, a wrapper for “char[]”
- Edit, delete, chars...

- Performance:
  - … usefull for list iteration + concatenation
  - Javac compiler already generate StringBuilder code

    => … useless for static “a” + “b” + 12 + …
- StringBuffer is deprecated (synchronized)

StringBuilder

```
StringBuilder buffer = new StringBuilder();
Collection<String> ls = Arrays.asList(new String[] { "text1", "text2" });
for(Iterator<String> iterator = ls.iterator(); iterator.hasNext();) {
    buffer.append(iterator.next());
    if (iterator.hasNext()) buffer.append(", ");
}
String text = buffer.toString();

StringBuilder buffer2 = new StringBuilder();
for(String elt : ls) buffer2.append(elt + ", ");
if (!ls.isEmpty()) buffer2.delete(buffer2.length()-2, buffer2.length());
if (!text.equals(buffer2.toString())) throw new IllegalStateException();
```

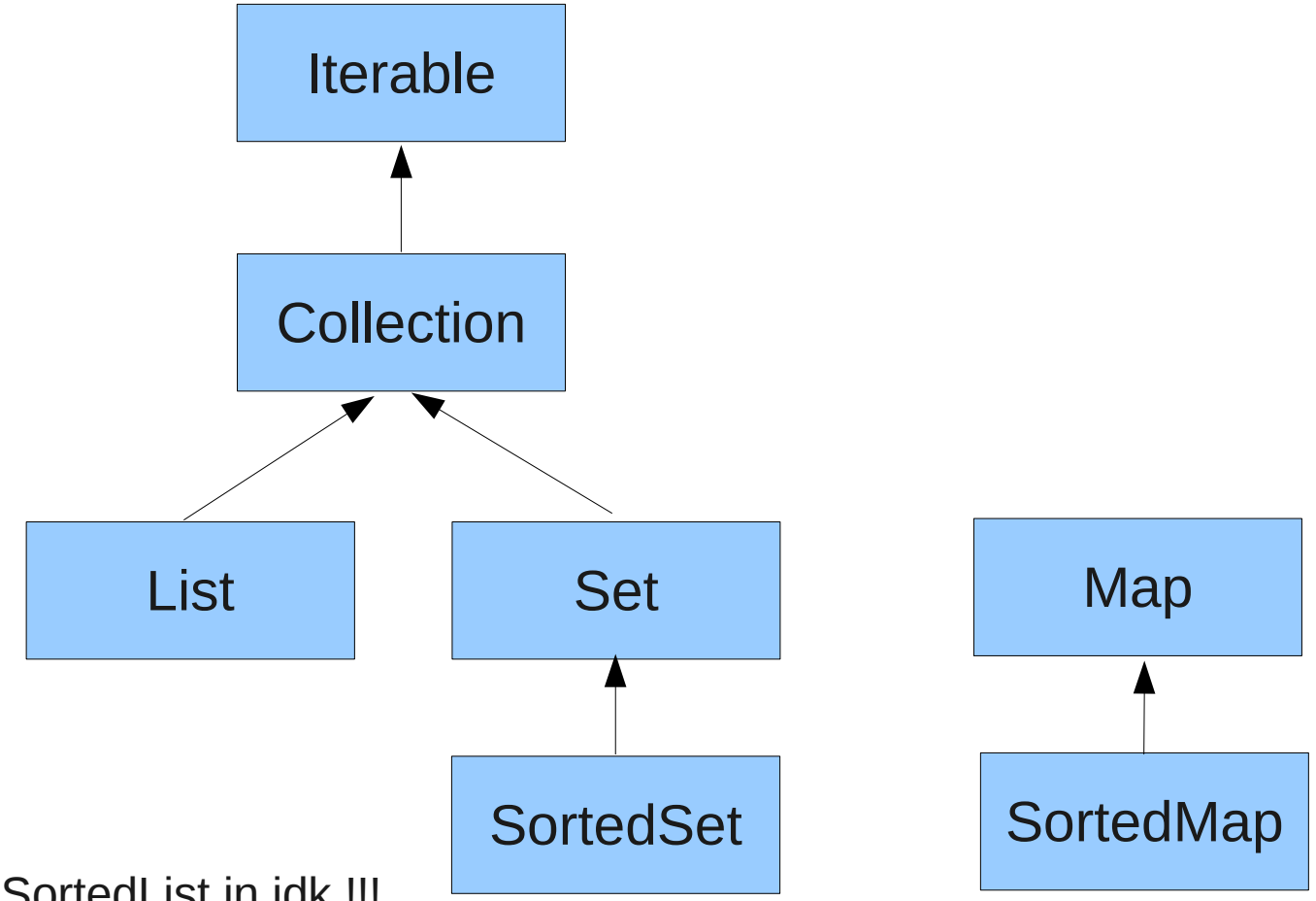
Java.util. Interfaces/Classes

- Java.util.* package contains both
  - Interface
    example: List
  - Simple implementations
    example ArrayList, LinkedList
  - Wrappers
    - Example: Collections.unmodifiableList(), Arrays.asList(), Collections.synchronizedList()
- See also
  - apache commons-collections, google guavac ...

Interfaces: Iterable,Collection,List,Set,Map

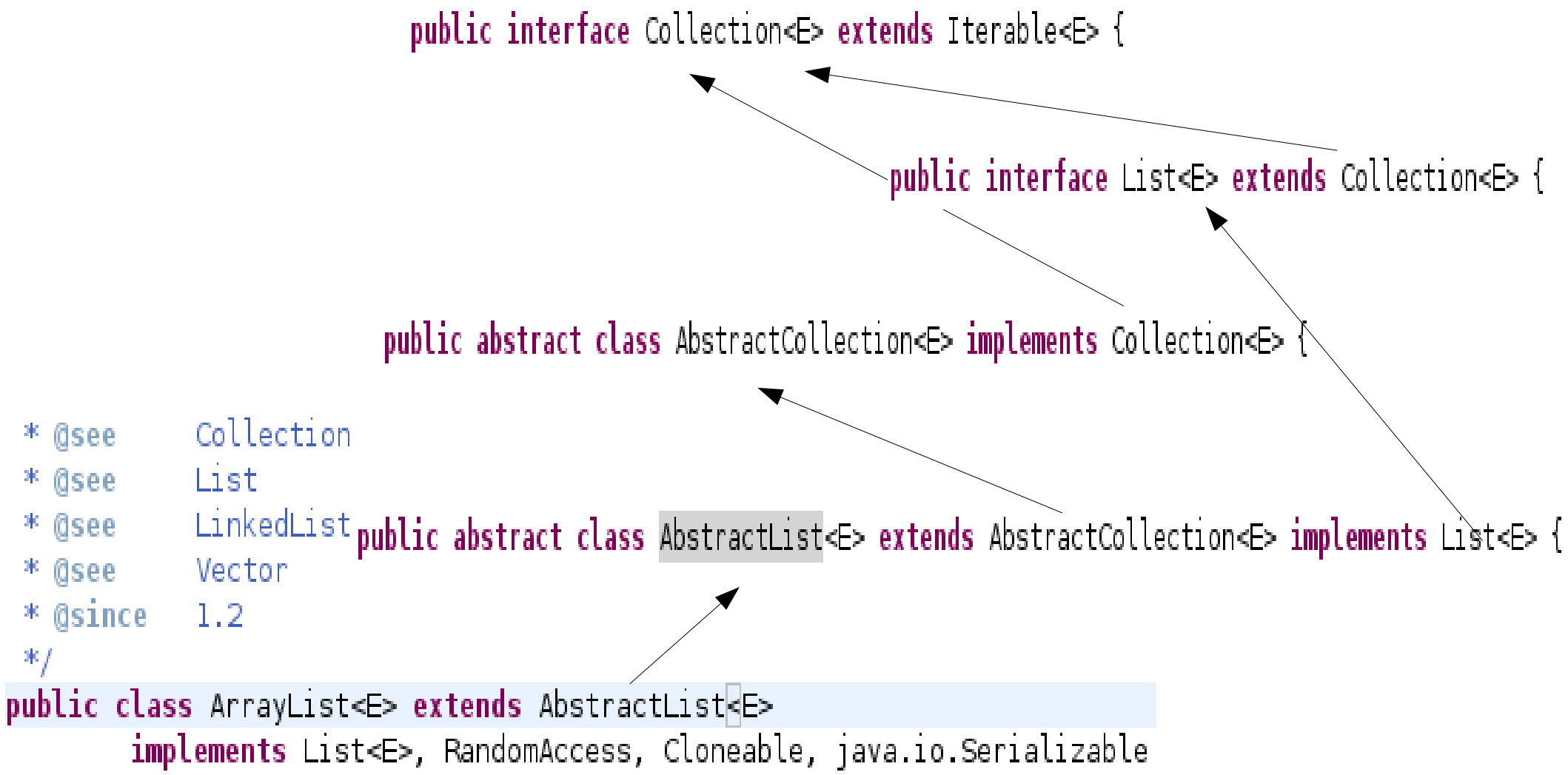
- Iterable<E> { public Iterator<T> iterator(); }
- Collection<E>
  - a mathematical bag, with no special order or property
  - add()/remove()/contains() for elements
    also clear()/addAll()/removeAll()/retainAll() …
    size()/isEmpty()/toArray()
- List<E> collection + index supports
- Set<E> collection + unicity
- Map<Key,E> unicity by key … =Set<Entry<K,V>>
- SortedSet, SortedMap

Interfaces



Note there is no SortedList in jdk !!!
=> cf google gavac for efficient ones

Java.util.ArrayList
Abstract classes and Interfaces



Sample ArrayList Code

```
List<String> ls = new ArrayList<String>();
ls.add("hello");

// add, remove, find by Object.equals() or by index
ls.add(1, "wor"); ls.set(1, ls.get(1) + "ld");
ls.add("."); if (ls.indexOf(".") == 2) ls.remove(2);

// Collection addAll/remove..
Collection<String> otherLs = Arrays.asList(new String[] { "text1", "text2" });
ls.addAll(otherLs);

// iterate by index, iterator, foreach
for(int i = 0; i < ls.size(); i++) System.out.println(ls.get((i)));
for (Iterator<String> iterator = ls.iterator(); iterator.hasNext();)
    System.out.println(iterator.next());
for(String elt : ls) System.out.println(elt);
```

Object Identity : equals()

- List / Collection only needs “equals()”
  for remove(),contains()...

- Equals is not “==”
  - Compare equality by VALUE
    .. not only by POINTER
  - Equals must be reflexive, symmetric, transitive...
  - == By default when not overridden (NOT RECOMMEDED)

- Equals may choose a subset of fields to compare
  - Example : “id” only

Sample Equals

```
public static class MyObj {
    int value1;
    String value2;

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        MyObj other = (MyObj) obj;
        if (value1 != other.value1) return false;
        if (value2 == null) {
            if (other.value2 != null) return false;
        } else if (!value2.equals(other.value2)) return false;
        return true;
    }
```

Typical List.remove() / indexOf() scan algorithms in o(N)

```
public boolean contains(Object o) {
    return indexOf(o) >= 0;
}
public int indexOf(Object o) {
    ListIterator<E> e = listIterator();
    if (o == null) {
        while (e.hasNext())
            if (e.next() == null)
                return e.previousIndex();
    } else {
        while (e.hasNext())
            if (o.equals(e.next()))
                return e.previousIndex();
    }
    return -1;
}
```

Special case for
“null.equals(elt)”
Return on first null

Linear iter
test equals() with elts
Return on first found

Theory => TP

1) Create a new java project in eclipse
2) Create a class with 4 id fields (an int, a double, a String and a bool), and other dummy fields
3) Implements the correct equals(), hashCode(), compareTo() for this class
4) Write a Junit test for playing with list add(),remove(),contains(), addAll()...

HASH table algorithms

- Doing search in O(N) is very inefficient
- Hash-Table offer an O(1) algorithm
  with memory consuption P
  - Choose P as prime number
  - with P >> N for avoiding conflicts

- to search an elt, compute its hash
  => search first in entry index “hash modulo P”
  check candidate with equals(),
  otherwise (conflict), check in next entry

Object.hashCode()

- HashCode() compute a HASH for an Object
  - When Object are equals =then=> hashCode equals
  - Reciprocity is false
- when not overriden (NOT RECOMMENDED), default hashCode() is the memory adress of the object …
  - The first time it was requested and stored !!!
    (gc can move objects)
  - Modulo 32bits for 64bits pointers

Sample hashCode()

```
public static class MyObj {
    int value1;
    String value2;

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + value1;
        result = prime * result + ((value2 == null) ? 0 : value2.hashCode());
        return result;
    }
```

Note on Hash Keys Immutability

- HashMap should contains readonly Keys!!
  class MyKey { private **final** String key1; …. }
  .. in doubt, do not modify objects after enlisted in maps!!!

```
public static class MyKey {
    private final int value1;
    private final String value2;

    public MyKey(int value1, String value2) {
        super();
        this.value1 = value1;
        this.value2 = value2;
    }

    @Override
    public int hashCode() {
```

Sample HashMap Code

HashTable is deprecated (synchronized), use HashMap

```
Map<MyKey,MyObj> map = new HashMap<MyKey,MyObj>();
map.put(new MyKey(123, "keyValue2"), new MyObj());
MyObj checkObj = map.get(new MyKey(123, "keyValue2"));
if (checkObj == null) throw new IllegalStateException();

// iterate over Map.Entry<K,E>
for(Map.Entry<MyKey,MyObj> e : map.entrySet()) {
    System.out.println(e.getKey() + " : " + e.getValue());
}
// Collection/Set views for entrySet(), keys(), values()
Set<Entry<MyKey, MyObj>> entries = map.entrySet();
Set<MyKey> keySet = map.keySet();
Collection<MyObj> values = map.values();

if (entries.size() != map.size() || keySet.size() != map.size()
        || values.size() != map.size()) throw new IllegalStateException();
```

HashSet

- Same as HashMap … key is also the value
- See internal implementation in jdk:

```
public class HashSet<E>
    extends AbstractSet<E>
    implements Set<E>, Cloneable, java.io.Serializable
{
    static final long serialVersionUID = -5024744406713321676L;

    private transient HashMap<E,Object> map;

    // Dummy value to associate with an Object in the backing Map
    private static final Object PRESENT = new Object();

    /**
     * Constructs a new, empty set; the backing <tt>HashMap</tt> instance has
     * default initial capacity (16) and load factor (0.75).
```

Sample HashSet Code

```
Set<MyKey> set = new HashSet<MyKey>();
set.add(new MyKey(123, "keyValue1"));
Set<MyKey> otherSet = new HashSet<MyKey>();
set.addAll(otherSet);
// remove()/contains()
if (!set.contains(new MyKey(123, "keyValue1"))) throw ne
set.add(new MyKey(456, "keyValue2"));
set.remove(new MyKey(456, "keyValue2"));
if (set.contains(new MyKey(456, "keyValue2"))) throw new

// iterate .. NO special order!
for(MyKey elt : set) {
    System.out.println(elt);
}
```

Red-Black Tree Algorithm

- well distributed Tree :
  N elements => depth log(N)
- On each node
  - A node contains a value … to compare with others
  - A node has 2 child left - right
  - All elements in left Tree are compared <= node
  - All elements in left Tree are compared > node
- Performance:
  - Operations in O(Log(N)) : add(), remove(), contains()...
  - Tree is ordered => Scan in sorted order... unlike hash

Sample compareTo() method

```
@Override
public int compareTo(MyKey obj) {
    Integer.valueOf(1).compareTo(2);
    if (this == obj) return 0;
    if (obj == null) return +1;
    if (value1 != obj.value1) {
        return (value1 < obj.value1)? -1 : +1;
    }
    int res = value2.compareTo(obj.value2);
    if (res != 0) return res;
    return 0;
}
```

Sample TreeMap code

- Same sample as HashMap …
  only change new HashMap() by new TreeMap()

```
Map<MyKey,MyObj> hashMap = new HashMap<MyKey,MyObj>();
// => use new TreeMap() instead of new HashMap()
Map<MyKey,MyObj> map = new TreeMap<MyKey,MyObj>();
// ... same Map interface => same sample code
```

- Specific samples for sorted key collections

```
// Collection/Set views for entrySet(), keys(), values()
Set<Entry<MyKey, MyObj>> entries = map.entrySet(); //<== sorted per key!
Set<MyKey> keySet = map.keySet();  //<== sorted!
Collection<MyObj> values = map.values(); // no special order for values

List<MyKey> sortedKeys = new ArrayList<MyKey>(map.keySet());
for(MyKey k : sortedKeys) System.out.println(k);
```

Sample TreeSet

- Same sample as HashSet …
- Specific code for sorting collections

```
SortedSet<MyKey> set = new TreeSet<MyKey>();
set.add(new MyKey(123, "keyValue1"));
set.add(new MyKey(5, "keyValue2"));
set.add(new MyKey(304, "keyValue3"));

List<MyKey> sortedList = new ArrayList<MyKey>(set);
for(MyKey elt : sortedList) {
    System.out.println(elt);
}

MyKey first = set.first(); System.out.println("first:" + first);
MyKey last = set.last(); System.out.println("last:" + last);
Set<MyKey> subSet = set.subSet(new MyKey(10, ""), new MyKey(200, ""));
System.out.println("subSet:" + subSet.size());
```

Theory => TP

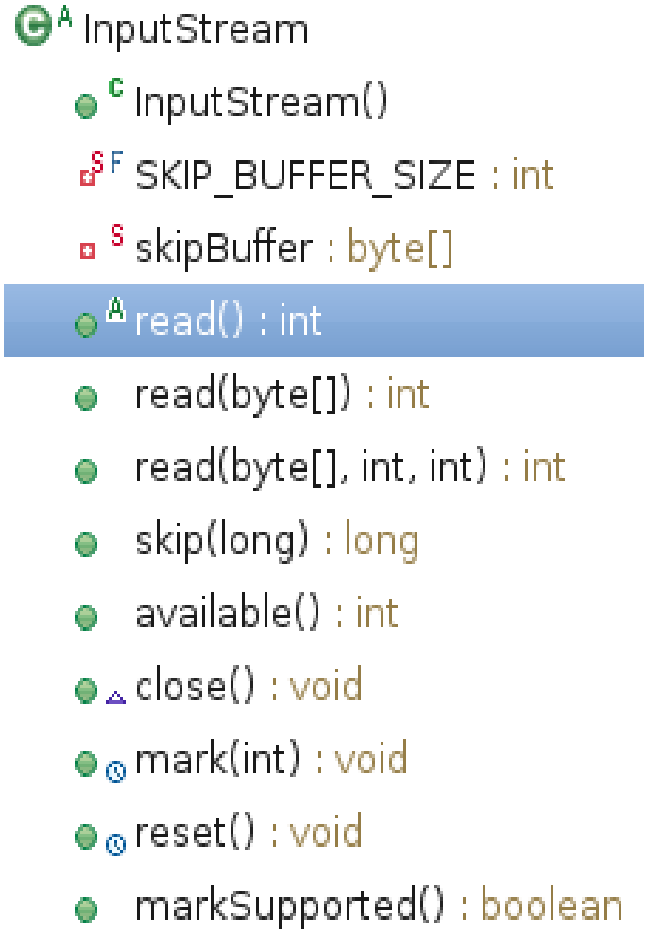
1) Recursive Scan in a directory file

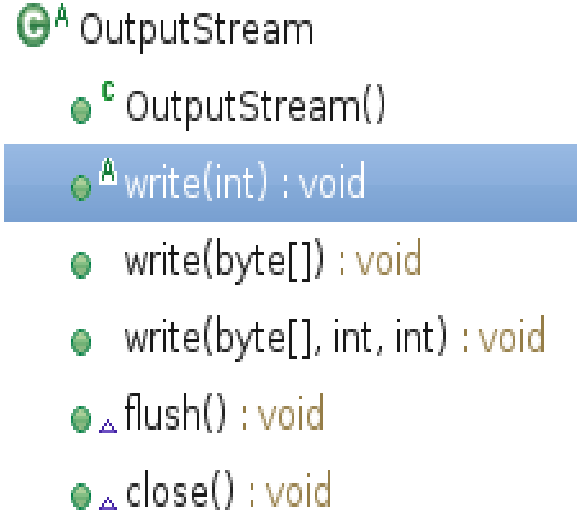
2) Filter out meaningless (hidden) files and sub-dirs starting with “.” name (.svn/*, .git, ...)

3) Count the number of files having the same name:
   Maintain a “Map<String,List<File>>” occurences.
   At the end, print the list of duplicates (count > 1) files names, with the list of corresponding occurences

Input/Output Stream Classes

- Stream is a way to handle bytes, one after one
  - Input => for reading byte(s)
  - Output => for writing byte(s)





Java.io. Input/Output Stream

- Stream are abstract classes .. see sub-classes
  - FileInputStream (resp Ouput)
    = based on an underlying java.io.File
  - BufferedInputStream (resp Ouput)
    = based on another Stream, plus a buffer
  - ByteArrayInputStream (resp Output)
    = based on an in-memory array
  - SocketInputStream (resp Ouput)
    = based on a Socket
  - StringBufferInputStream … deprecated!!

Sample ByteArrayInputStream

```
byte[] bytes = new byte[] { 0, 1, 2, 3 };
InputStream in = new ByteArrayInputStream(bytes);

int b0 = in.read(); assert 0 == b0; // read 1 byte
int b1 = in.read(); assert 1 == b1; // read 1 next byte
byte[] b23 = new byte[2];
in.read(b23); // read an array of 2 next bytes
assert 2==b23[0] && 3==b23[1];
```

Typical code with try-finally close and IOException

```
InputStream inStream = null;
try {
    inStream = new BufferedInputStream(new FileInputStream(inputFile));
    for(;;) {
        int b = inStream.read();
        if (b == -1) break; // End-Of-File
        // handle byte...
    }
} catch(IOException ex) {
    throw new RuntimeException("Failed to read file", ex);
} finally {
    if (inStream != null) try { inStream.close(); } catch(Exception ex) {}
}
```

Sample Code: InputStream => SHA1 digest (as in git hash-object)

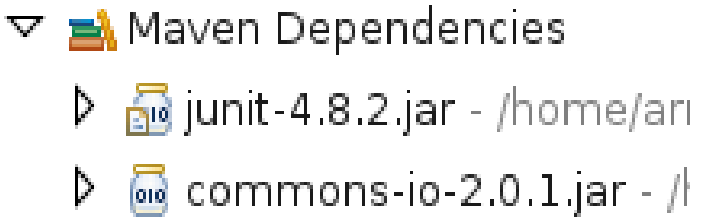
```
MessageDigest sha1 = MessageDigest.getInstance("SHA1");
// add git BLOB prefix in SHA1, to compare with $  git hash-object file
sha1.update(("blob " + inputFile.length()).getBytes());
sha1.update((byte) 0);

InputStream inStream = null;
try {
    inStream = new BufferedInputStream(new FileInputStream(inputFile));
    for(;;) {
        int b = inStream.read();
        if (b == -1) break; // End-Of-File
        sha1.update((byte) b);
    }
} catch(IOException ex) {
    throw new RuntimeException("Faile to read/write file", ex);
} finally {
    IOUtils.closeQuietly(inStream);
}
byte[] sha1Digest = sha1.digest();
String sha1Str = Hex.encodeHexString(sha1Digest);
```

Sample Copy Input->Output Code

```
File inputFile = new File("in.txt");
File outputFile = new File("out.txt");
InputStream inStream = null;
OutputStream outStream = null;
if (!inputFile.exists() || !inputFile.isFile() || !inputFile.canRead()) {
    throw new RuntimeException("can not read from file:" + inputFile);
}
if (!outputFile.exists() || (outputFile.isFile()? !outputFile.canWrite() : fa
    throw new RuntimeException("can not write to file:" + outputFile);
}
try {
    inStream = new BufferedInputStream(new FileInputStream(inputFile));
    outStream = new BufferedOutputStream(new FileOutputStream(outputFile));
    for(;;) {
        int b = inStream.read();
        if (b == -1) break; // End-Of-File
        outStream.write(b);
    }
} catch(IOException ex) {
    throw new RuntimeException("Faile to read/write file", ex);
} finally {
    if (inStream != null) try { inStream.close(); } catch(Exception ex) {}
    if (outStream != null) try { outStream.close(); } catch(Exception ex) {}
}
```

Same with Jakarta commons-io !!

```xml
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.0.1</version>
</dependency>
```




```java
File inputFile = new File("in.txt");
File outputFile = new File("out.txt");
try {
    FileUtils.copyFile(inputFile, outputFile);
} catch (IOException ex) {
    throw new RuntimeException("Failed to copy file", ex);
}
```

Theory => TP

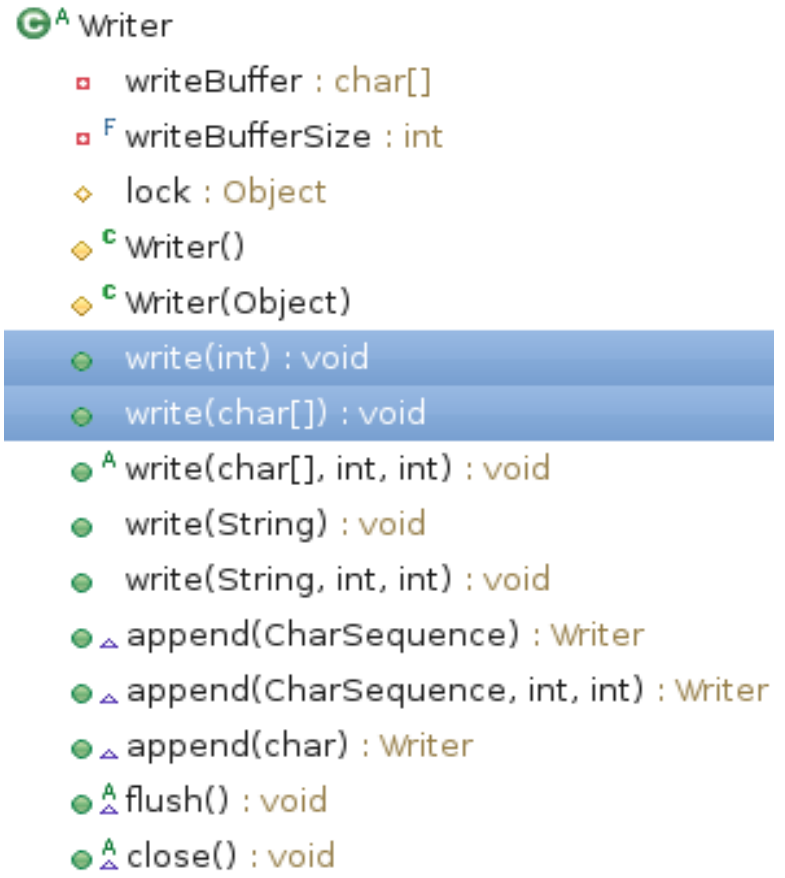
- Similar to previous exercise:

  Implements the search duplicate file by content instead of duplicate by filename

  - Use MessageDigest to compute SHA1 hashcode
  - Maintain a Map<SHA1,List<File>>
  - Print duplicates file contents

Reader/Writer abstract classes

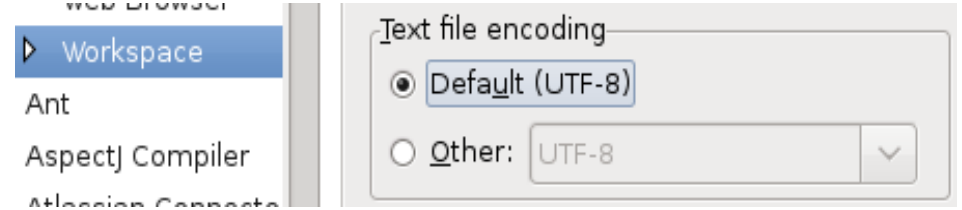
- Reader/Writer is a way to handle chars (not bytes), one after one
  - Reader => for reading char(s)
  - Writer => for writing char(s)



CharSet encoding/decoding

- Byte to char conversion is called **CharSet** Encoding/Decoding
- Char can be 1,2,3 or 4 bytes long in UTF-8
- standards ASCI chars are 1 byte ONLY !! (0 followed by 7 bits)
- Other standard Encoding: ISO-8859-1=Latin1, Cp1253=Windows

UTF-8 / ISO-8859-1 nightmare ... English Javadoc



file(s), viewed as UTF-8 file

```
/**
 * French has a lot of accents: à é è ê
 * Things usually goes wrong as this...
 * French has a lot of accents: Ã  Ã© Ã¨ Ãª
 * Of course, all javadoc comments should be written in English (no accent!)
 * Java itself is compiled into UTF-8 bytecode
 */
```

Line written in ISO-8859-1

EXACTELY SAME file, viewed as ISO-8859-1 file

```
/**
 * French has a lot of accents: ? ? ? ?
 * Things usually goes wrong as this...
 * French has a lot of accents: à é è ê
 * Of course, all javadoc comments should be written in English (no accent!)
 * Java itself is compiled into UTF-8 bytecode
 */
```

Line written in UTF-8

InputStreamReader Sample Code

- understand class behavior
  by reading camel-case classname from right to left ...

  Input Stream Reader =
  a Reader sub-class adapter
  based on InputStream

```
byte[] helloWorldUTF8Bytes = "Hello World".getBytes(Charset.forName("UTF8"));
InputStream inStream = new ByteArrayInputStream(helloWorldUTF8Bytes);
// convert InputStream (byte) to Reader (char) using UTF8 encoding
Reader reader = new InputStreamReader(inStream, Charset.forName("UTF8"));
int ch = reader.read();
int ch1 = reader.read();
if ('H' != (char)ch || 'e' != (char)ch1) throw new IllegalStateException();
```

Sample BufferedReader Code

- Perfect class for reading line per line

```
Reader inReader = new InputStreamReader(System.in);
BufferedReader lineReader = new BufferedReader(inReader);
String line;
for(;;) {
    System.out.println("prompt> ");
    line = lineReader.readLine();
    if (line == null) { break; } // End-Of-File  (Control+D)
    System.out.println("ok, read line: '" + line + "'");
}
System.out.println("EOF");
```

```
prompt>
test
ok, read line: 'test'
prompt>

ok, read line: ''
prompt>
EOF
```

PrintStream class

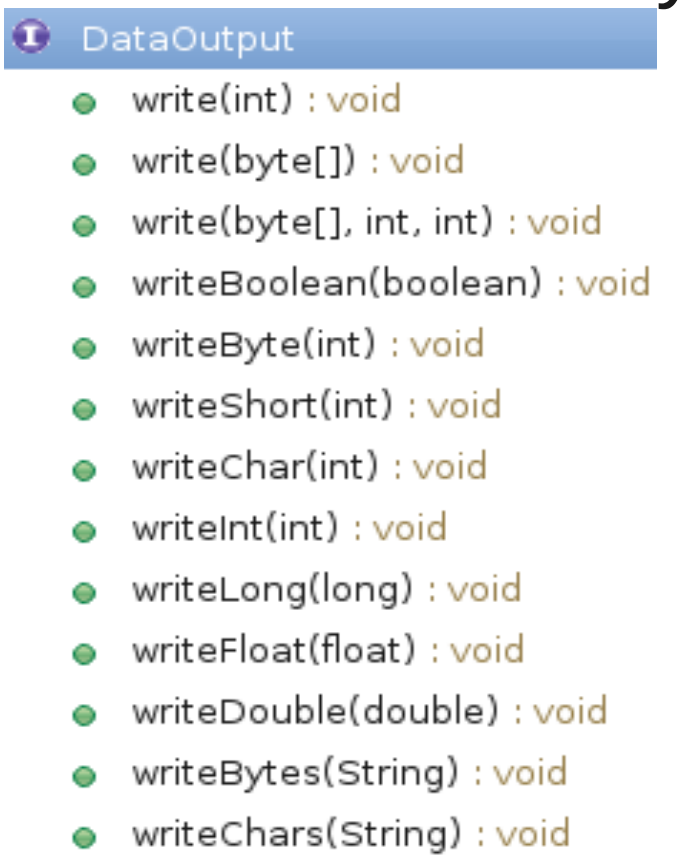
- Extension of OutputStream class, for formating int,long,bool...as String, and printing newline
- Class used in “System.out” static field

```
public class PrintStream extends FilterOutputStream
    implements Appendable, Closeable
{

    private boolean autoFlush = false;
    private boolean trouble = false;
    private Formatter formatter;

    /**
     * Track both the text- and character-output streams, so t
     * can be flushed without flushing the entire stream.
     */
    private BufferedWriter textOut;
    private OutputStreamWriter charOut;
```

- newLine() : void
- print(boolean) : void
- print(char) : void
- print(int) : void
- print(long) : void
- print(float) : void
- print(double) : void
- print(char[]) : void
- print(String) : void
- print(Object) : void
- println() : void
- println(boolean) : void
- println(char) : void
- println(int) : void
- println(long) : void
- println(float) : void
- println(double) : void
- println(char[]) : void
- println(String) : void
- println(Object) : void

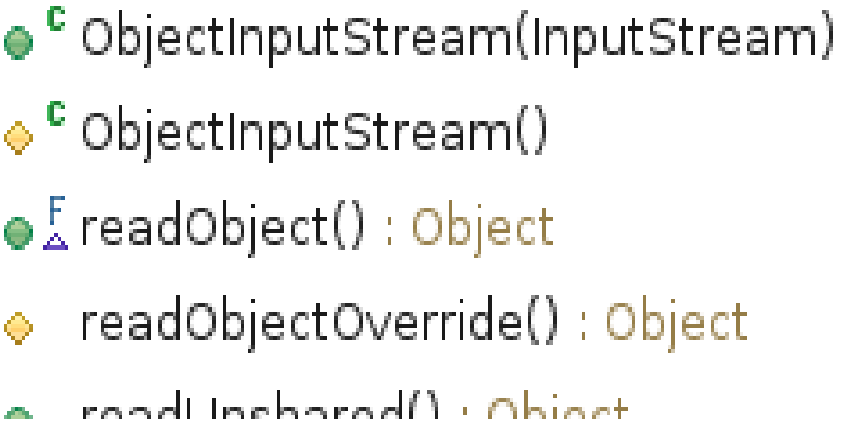
Data Input/Output Stream

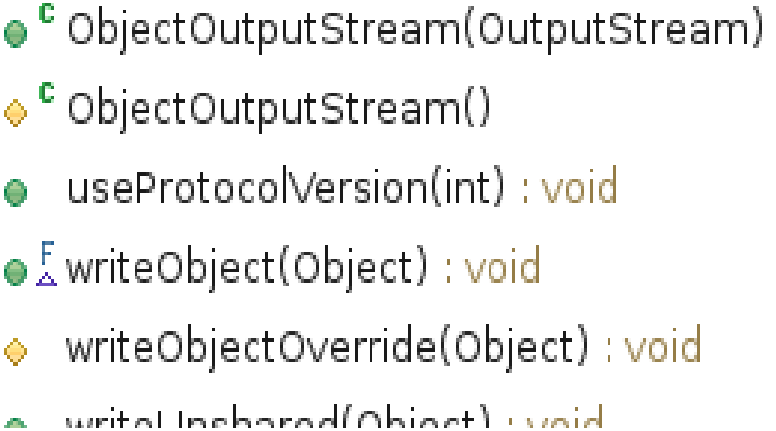
- Class for serializing int,long,double,String ... as bytes to underlying Input/Output Stream (NOTE: interface = Data Input/Output)
- for reading/writing compressed data manually



Object Input/Output Stream

- Magic Serialization using Java Reflection
- Used to save objects to file / to RMI sockets...
- Object =(serialize)=> bytes =(deserialize)=> Object





Implements java.io.Serializable

- Serializable is an EMPTY interface
  … used as a marker
  to check if object are to be serialized
- Read is checking compatibility signatures...
  cf serialVersionID (HASH for fields + methods !!)

```
public static class MySerializableObj implements java.io.Serializable {

    private static final long serialVersionUID = 1L; // useless..so force in java.io.Serializable

    private int field1;
    private transient int unserializedField2;
```

Sample Object Input/Output Stream

```
MySerializableObj obj = new MySerializableObj();
ObjectOutputStream oout = null;
try {
    oout = new ObjectOutputStream(new FileOutputStream("myobj.ser"));
    oout.writeObject(obj); // serialize Object => file
} finally {
    oout.close();
}

MySerializableObj obj2;
ObjectInputStream oin = null;
try {
    oin = new ObjectInputStream(new FileInputStream("myobj.ser"));
    obj2 = (MySerializableObj) oin.readObject(); // deserialize file => new Obj
} catch (ClassNotFoundException ex) {
    throw new IllegalStateException("should not occur", ex);
} finally {
    oin.close();
}
if (obj2 == obj || !obj2.equals(obj)) throw new IllegalStateException();
```

New package java.nio.*

- NOT Complex enough ??? Don't Worry

- You can have **InputChannel** / **OutputChannel**
  and **Buffer (ByteBuffer, DirectBuffer, HeapBuffer...)**

  Same as InputStream / OutputSteam
  but data copy per buffer blocks (<=4096...),
  not per bytes

  … for optimizing asynchronous DMA ios
  (less CPU consuption, better hardware Drivers resources)

Conclusion …. Beautiful java.* packages...
MUST BE KNOWN

Questions ?

arnaud.nauwynck@gmail.com