A Swift Introduction to Java

Gevorderde Programmeertechnieken

Prof. Dr. Steven Latré

Universiteit Antwerpen

Hello World!

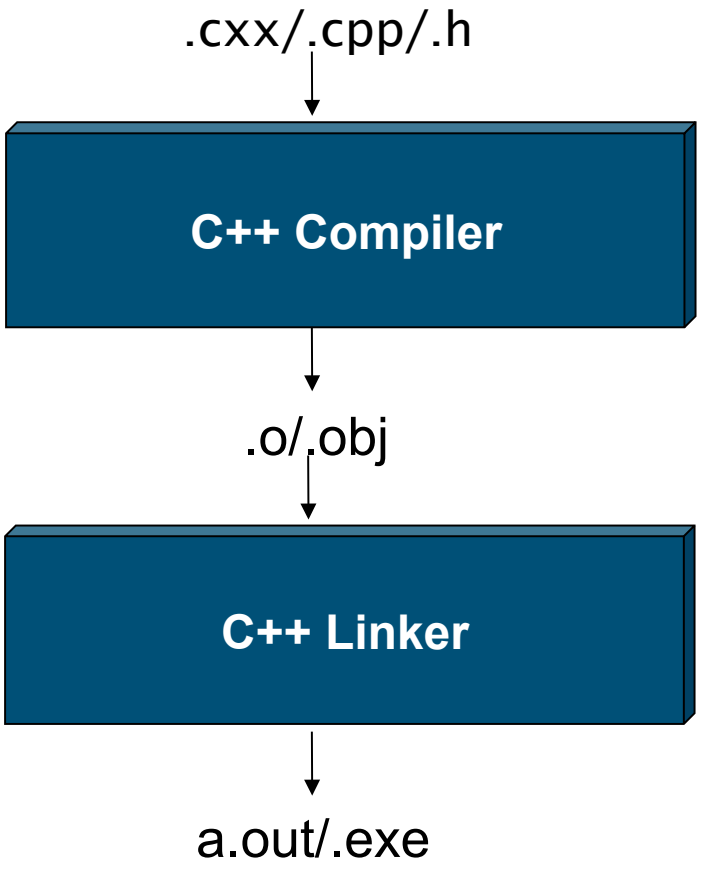
C++

```
int main () {
      std::cout << “Hello World!” << std::endl;
      return 0;
}
```

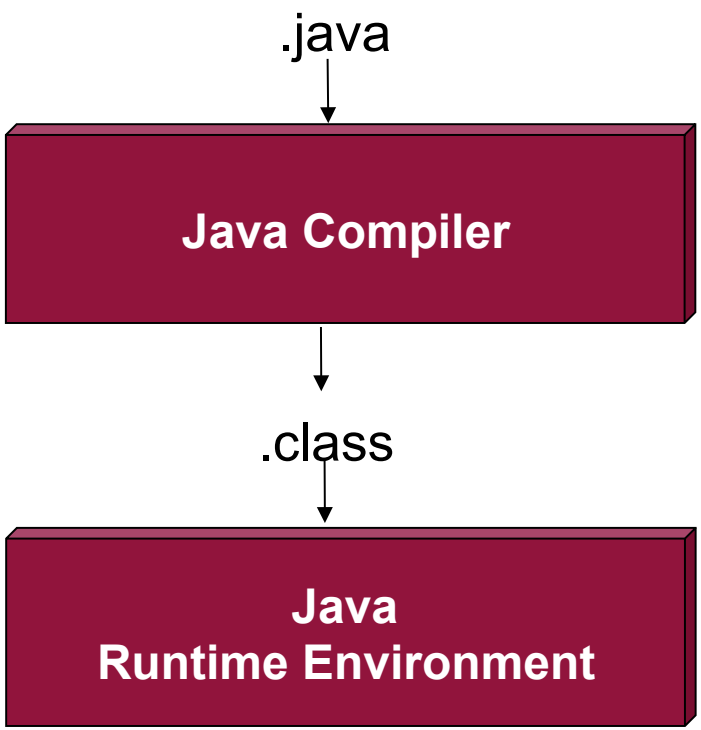
Java

```
public class HelloWorld {
      public static void main(String[] args) {
         System.out.println(“Hello World!”);
      }
}
```

Compilation

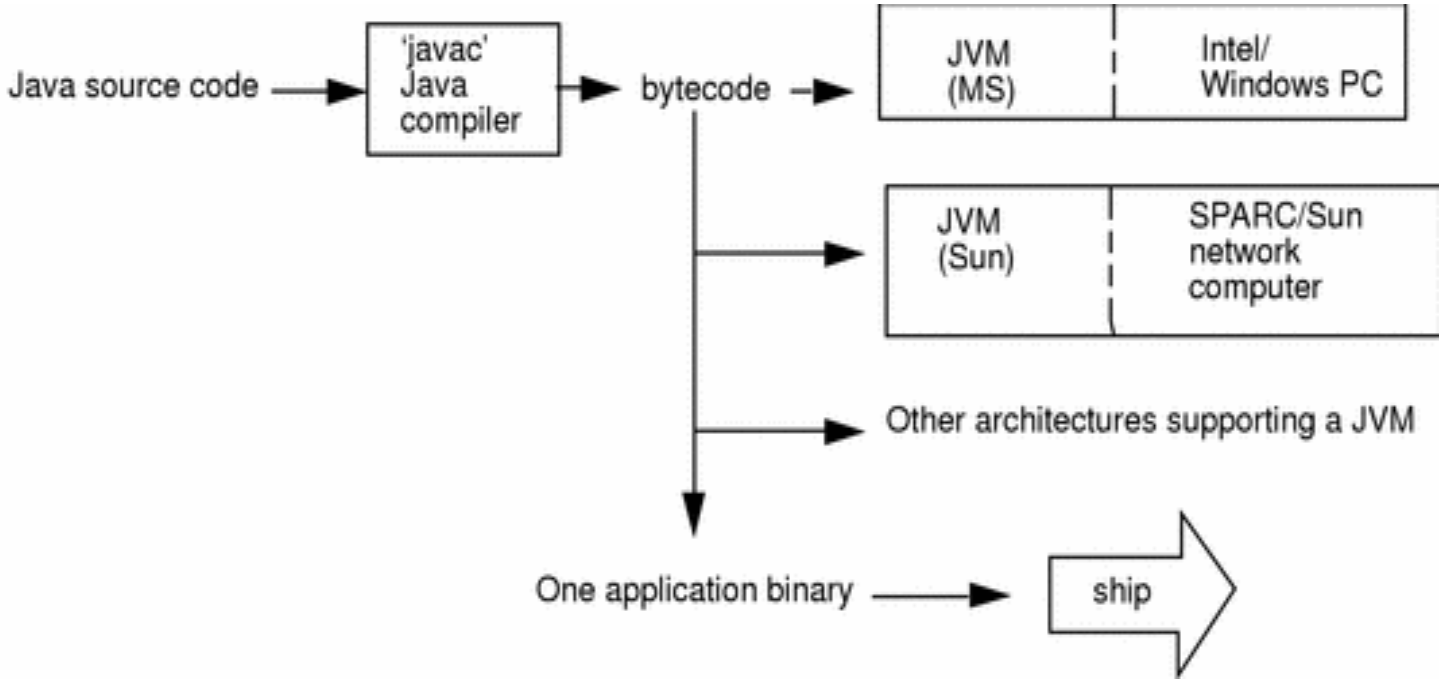


$>./a.out
Hello World!



$>java HelloWorld
Hello World!

Compilation



Sample program

Universiteit Antwerpen

```
1  package myPackage;
2
3  public class Point2D extends Object implements IPoint2D, Comparable {
4      private double x;
5      private double y;
6      private static double epsilon = 0.00002;
7
8      public static void setEpsilon(final double epsilon) {
9          Point2D.epsilon = epsilon;
10     }
11
12     public Point2D(final double x, final double y) {
13         this.x = x;
14         this.y = y;
15     }
16
17     public Point2D(Point2D p) {
18         x = p.x;
19         y = p.y;
20     }
21
22     public double getX() {
23         return x;
24     }
25
26     public void setX(final double x) {
27         this.x = x;
28     }
29
30     public double getY() {
31         return y;
32     }
33
34     public void setY(final double y) {
35         this.y = y;
36     }
37
38     public String toString() {
39         return new String("(" + x + ", " + y + ")");
40     }
41
42     public boolean equals(Object o) {
43         if (this == o)
44             return true;
45         if (!(o instanceof Point2D))
46             return false;
47         Point2D p = (Point2D) o;
48         return (Math.abs(x - p.x) < Point2D.epsilon) && (Math.abs(y - p.y) < Point2D.epsilon);
49     }
50
51     public int compareTo(Object p) {
52         if (this.equals(p))
53             return 0;
54         if (y > ( (IPoint2D)p ).getY()) {
55             return 1;
56         }
57         return -1;
58     }
59 }
```

Interfaces & abstract classes

Universiteit Antwerpen

Interface `IPoint2D`

```
1  package myPackage;
2
3  interface IPoint2D {
4      void setX(double px);
5      double getX();
6      void setY(double py);
7      double getY();
8  }
```

`.abstract` keyword

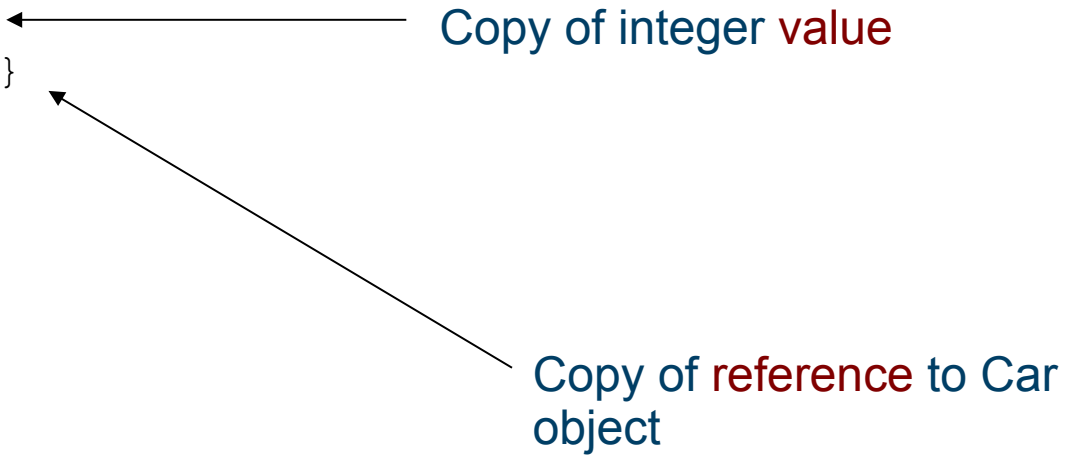
```
3  public abstract class MyAbstractClass {
4      private int field;
5
6      abstract void foo();
7  }
```

Parameter passing

All types are passed by value

- Primitive type (non-reference type)
  - int, long, float, double, char, boolean, ...
- Non-primitive types (reference type)
  - Double, double[], MyCoolClass, ...

```
void myFunc(int b) {}
void myFunc(Car car) {}
```



Memory Management

No user-managed memory
Automatic garbage collection
- Reference counting
- Unpredictable

GC calls finalize method
On request : System.gc()

"When control returns from the method call, the Java Virtual Machine has made a best effort to reclaim space from all discarded objects" (Sun Javadoc)

Max. heap size tweaking
java -Xmx1024M -Xms256M ...

Access control

Keywords

- `public:` all
- `protected:` subclasses
- `private:` none
- `Default:` arial
- `package access :` every class in the package

Control flow

```
24          for(int i = 0; i < 10; i++) {
25
26          }
27
28          do {
29
30          } while (cond);
31
32
33          while(cond) {
34
35          }
36
37          if(cond) {
38
39          } else if(cond) {
40
41          } else {
42
43          }
44
45
46          switch(myVar) { //byte, short, char, int, enum
47              case 2 : System.out.println("myVar 2"); break;
48              case 5 : System.out.println("myVar 5"); break;
49              default: System.out.println("Default clause"); break;
50          }
51
```

Control flow

```
33          while(cond) {
34              //...
35              continue; //Move to next iteration
36              //...
37          }
38
39          while(cond) {
40              //...
41              break; //Break out of loop
42              //...
43          }
```

Object class

Superclass of every Java class

- equals() / hashCode() / toString() / ...

Object clone() method

Field-by field shallow copy (protected access)

Deep copy on class A

- Override clone() method
- Make public
- Call super.clone() and catch CloneNotSupportedException
- Add deep copy semantics
- Implement Cloneable *tagging* interface
- Warning! Implies cloneability of all derived classes

Collections

- Framework
  - Interfaces
  - Implementations
  - Algorithms (Collections)
- Collection and Map interfaces
  - Iteration
  - Size calculation
  - Element addition / removal

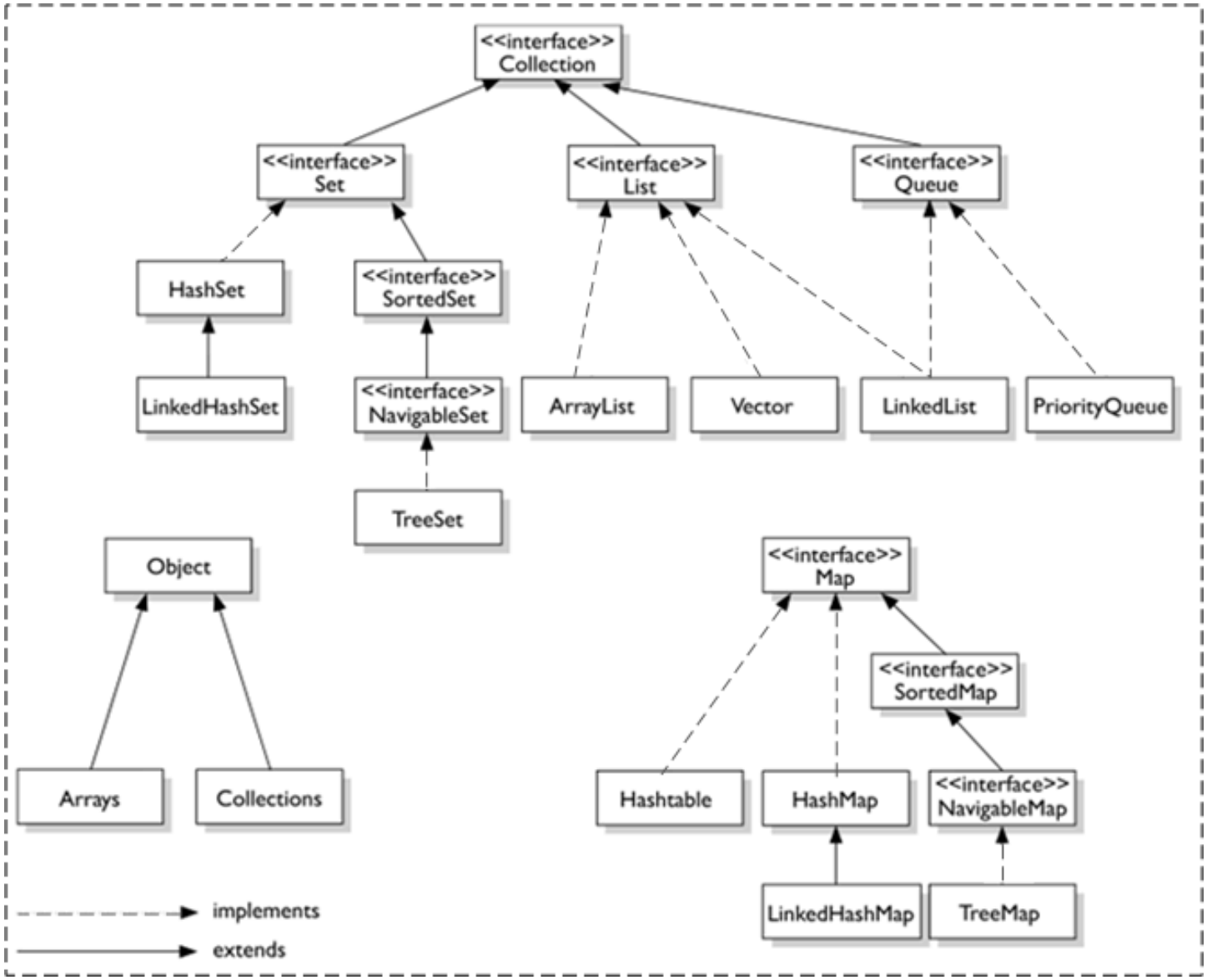
...

- Object-based, primitives are *auto-boxed*
- Collections.sort()

int compareTo() (implement Comparable interface)

- http://download.oracle.com/javase/tutorial/collections/

JCF overview



Raw arrays

```
32          int[] myArray = {2, 3, 5, 7};
33          int[] copy = Arrays.copyOf(myArray, myArray.length);
34          Arrays.sort(myArray);
35
36          int[][] array2D = new int[rows][cols];
```

Exceptions

```java
/**
 * Method that implements the division
 * @param x dividend
 * @param y divisor, should be != 0
 * @return x / y
 * @throws DivisionByZeroException when y = 0
 */
public static int divide (int x, int y) throws DivisionByZeroException {
    if (y == 0) throw new DivisionByZeroException ();
    return x / y;
}

/**
 * Method that uses the division and makes a DivisionByZeroException thrown
 */
public static void useDivide () throws DivisionByZeroException {
    divide(5,0);
}
```

Exceptions: stacktrace

```
/**
 * Main method that tests the divide-method
 * @param args commandline arguments
 */
public static void main(String[] args) {
    int dividend = 20;
    int divisor[] = {0, 1, 2, 3, 4, 5, 6};

    for (int i = 0; i < divisor.length; i++) {
        System.out.println("Dividend: " + dividend);
        System.out.println("Divisor : " + divisor[i]);

        try {
            System.out.print("Dividend/Divisor: ");
            System.out.println(Division.divide(dividend, divisor[i]));
            System.out.println();
        } catch (DivisionByZeroException e) {
            System.out.println("Exception catched: " + e.getMessage());
            e.printStackTrace();
        }
        System.out.println();
    }
}
```

Problems | Javadoc | Declaration | Package Explorer | Console

<terminated> Session2.Exercise3 [Java Application] /usr/java/jdk1.5.0_06/bin/java (13-feb-2006 10:23:52)

```
Dividend: 20
Divisor : 0
Dividend/Divisor: Exception catched: Error: Division by zero!
be.ac.ua.comp.lvgompel.vb.ex.exceptions.DivisionByZeroException: Error: Division by zero!
        at be.ac.ua.comp.lvgompel.vb.ex.math.Division.divide(Division.java:27)
        at be.ac.ua.comp.lvgompel.vb.ex.cppSession2.Exercise3.main(Exercise3.java:36)
```

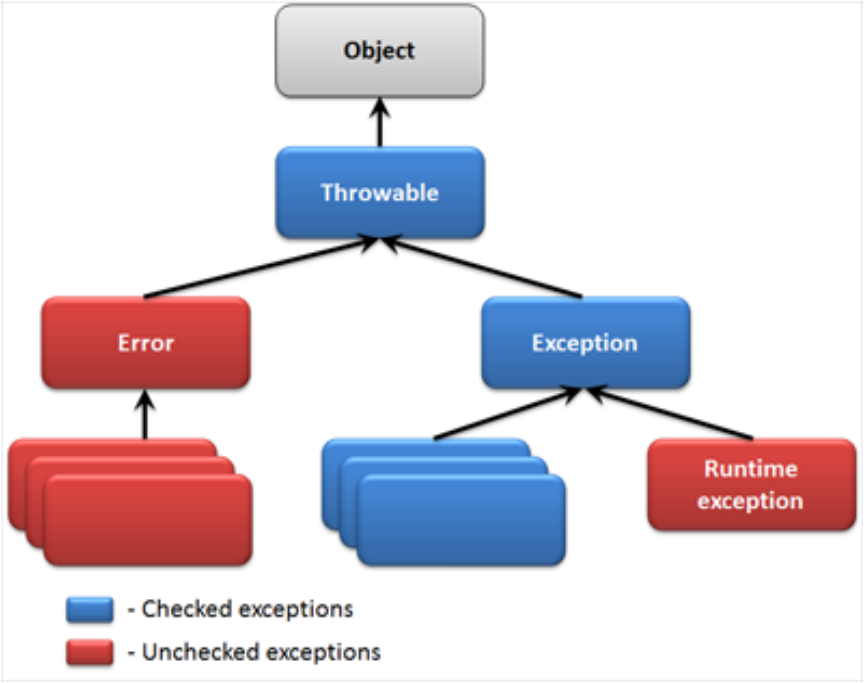
Exceptions

Types

RuntimeException (unchecked)

Bad cast / Division by zero / Out-of-bounds array access

Exception (checked)



Handling

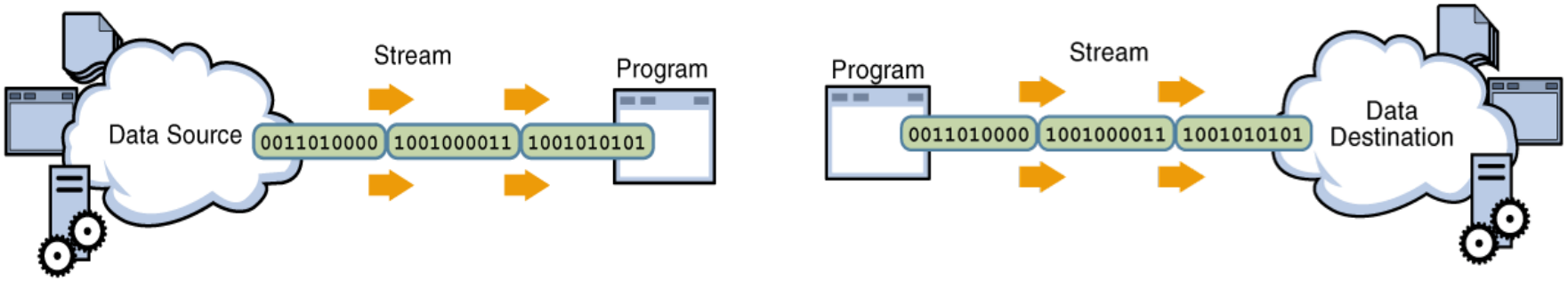
- try {}
- catch {}
- finally{}

Guidelines

- Define exception types to enable tailored catch clauses
- Throw the right types
- Do not expose implementation
- Do not throw too many different types (<4)
- Do not squelch!

I/O

Stream-based model




```
System.in / System.out vs cin/cout
BufferedInputStream s = new
  BufferedInputStream(new
  FileInputStream(new
    File(“”)));
```

- Streams need to be closed with close()
- More than 60 stream types available

http://download.oracle.com/javase/tutorial/essential/io/index.html

Streams

InputStream (byte ops)
FileInputStream (byte ops file)
BufferedInputStream (buffering)
DataInputStream (primitive types)
ObjectInputStream (objects)

- Class of object involved in I/O must implement Serializable

Unicode I/O
Reader / Writer hierarchy
Input: BufferedReader
Output: PrintWriter

Reflection

Class

Obtain

- Object.getClass()
- boolean.class
- Class.forName(“myPackage.myClass”)

getModifiers / getGenericInterfaces() / getField(s)() / getConstructor(s)() / getMethod(s)()
isAssignableFrom(Class toAssign)

Constructor

c.newInstance()
Prefer over newInstance() on Class

Field

f.setLong(obj, 20);

Method

m.invoke(obj, par1, par2, par3);

Miscellaneous

Strings

For large strings, use StringBuffer for concatenation instead of “ho” + “mer”
StringTokenizer

Annotations

@Deprecated / @Override / @SuppressWarnings

Javadoc

@author / @param / @return / @throws / @see / …