

PROGRAMMING IN



Today

- Java API
 - java.util
 - java.io
- More OOP
 - Generics
 - Enum
- .jar files
- Q & A



Announcements

Course website: <http://stuff.mit.edu/iap/2009/java/>

Email: sipb-iap-java@mit.edu

Wake up and smell the coffee!

Software

Java Development Kit (JDK) - <http://java.sun.com/javase/downloads/index.jsp>

Eclipse Platform - <http://www.eclipse.org/>

Reference

The Java Tutorial - <http://java.sun.com/docs/books/tutorial/index.html>

Java Language API - <http://java.sun.com/javase/reference/api.jsp>

Java SE Documentation - <http://java.sun.com/javase/downloads/index.jsp>

Java SE Source Code - <http://java.sun.com/javase/downloads/index.jsp>

Sun Microsystems

Not your daddy's cup of Joe...

There were five primary goals in the creation of the Java language:

- 1) It should be "simple, object oriented, and familiar".
- 2) It should be "robust and secure".
- 3) It should be "architecture neutral and portable".
- 4) It should execute with "high performance".
- 5) It should be "interpreted, threaded, and dynamic".

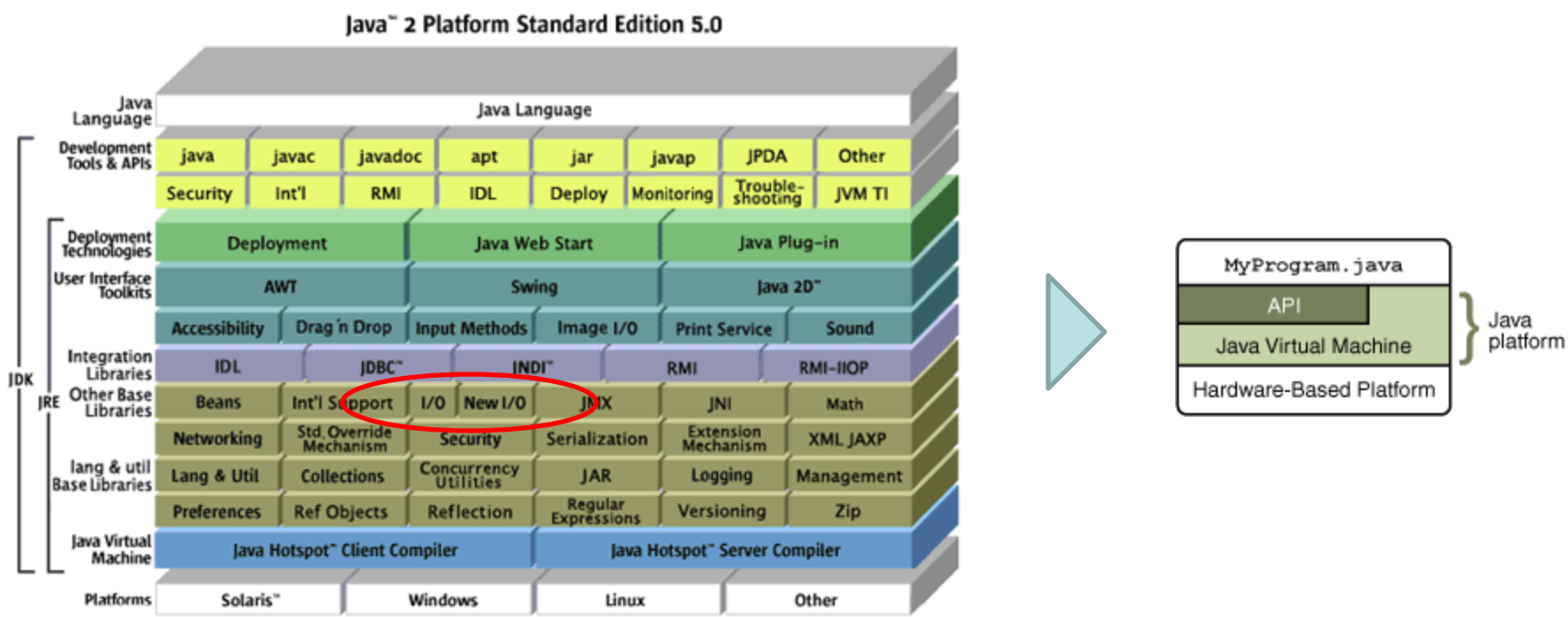
Java was designed to be **safe**, **simple**, and **powerful**.

JAVA API

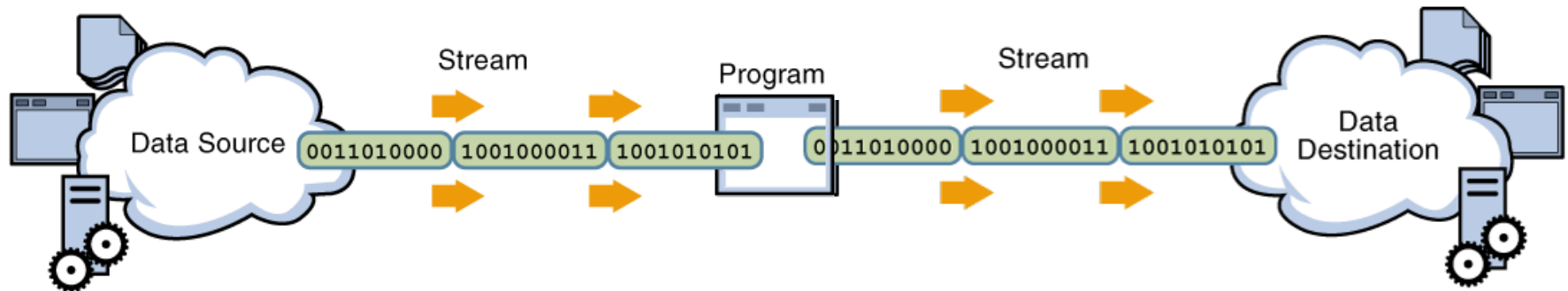


java.io

package java.io



I/O Streams



- socket
- file
- database
- byte array

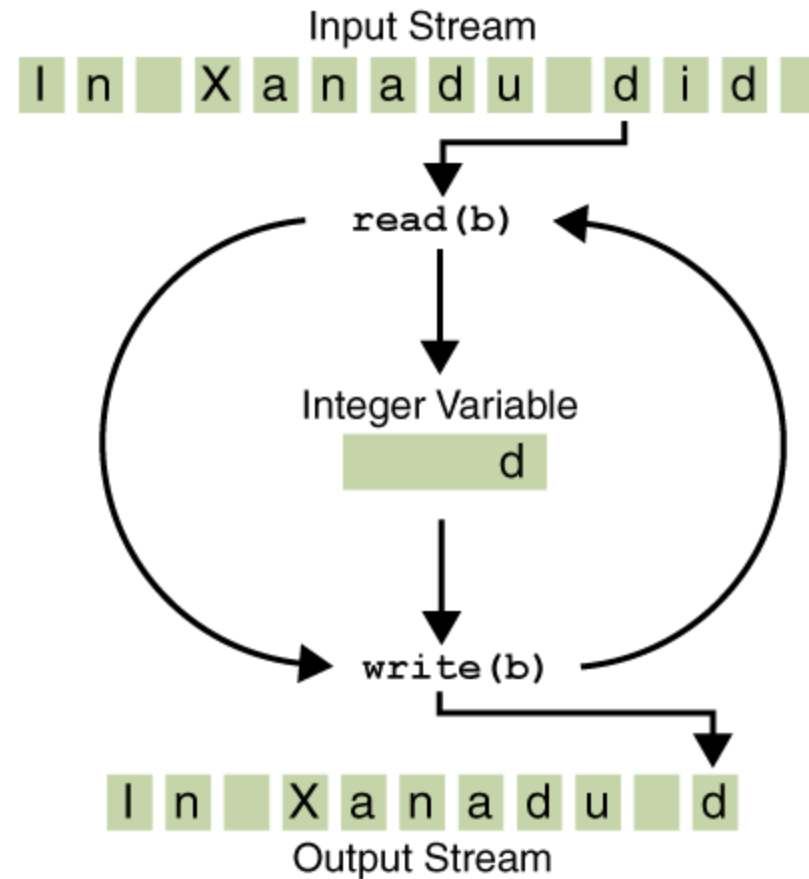
process data

- transform
- split
- buffer

- socket
- file
- database
- byte array

```
// Always close your streams!  
try {  
    // doProcessing  
} finally {  
    stream.close();  
}
```

Byte streams



Examples: CopyBytes.java, CopyCharacters.java, CopyLines.java

Other kinds of I/O

Standard:

- `System.in`
- `System.out`
- `System.err`

`java.net.Socket`:

- `socket.getInputStream()`
- `socket.getOutputStream()`

Serialization

- `java.io.ObjectOutputStream`
- `java.io.ObjectInputStream`



Interacting with the user

```
// construct a java.io.BufferedReader
BufferedReader in = new BufferedReader(
    new InputStreamReader(System.in));

in.readLine();

// CAREFUL! This only exists when java is started from the command line
Console c = System.console();
c.readLine();
char[] passwd;
if ((passwd = c.readPassword("[%s]", "Password:")) != null) {
    // verify password
    java.util.Arrays.fill(passwd, ' ');
}

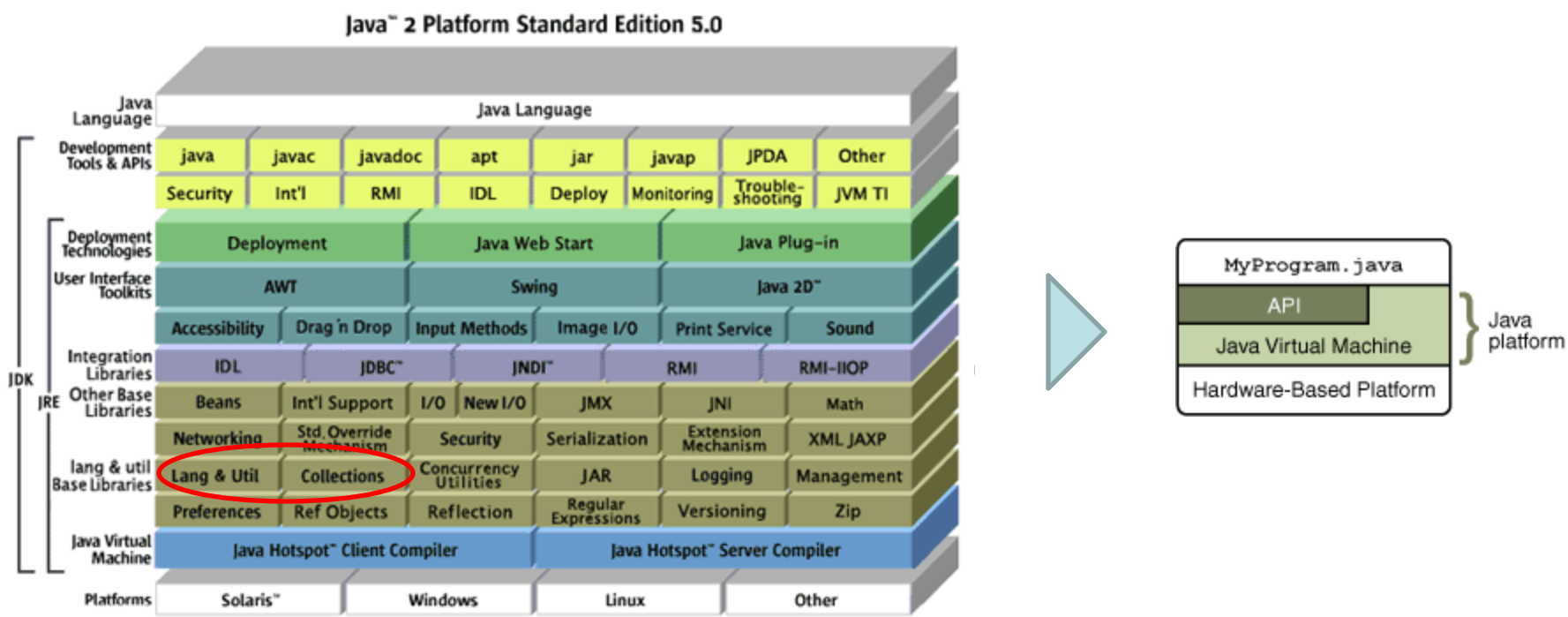
// use java.util.Scanner
Scanner s = new Scanner(System.in);
s.nextLine();
s.next(); // next parsing token (word, by default)
```

JAVA API



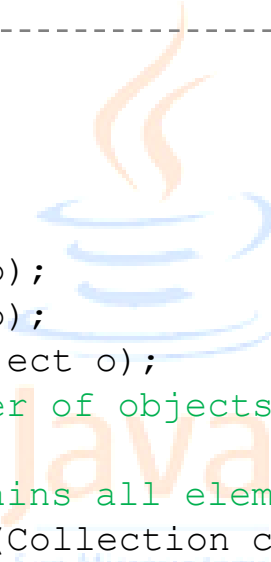
java.util

package java.util



Collection

At the most abstract level, a group of Objects.

A large, semi-transparent watermark of the Java logo is centered in the background of the code block. It features a stylized blue cup with three orange flames rising from it, and the word "Java" in a large, orange, serif font below it.

```
package java.util;

public interface Collection {

    public boolean add(Object o);
    public void remove(Object o);
    public boolean contains(Object o);
    public int size(); // number of objects

    // returns true if it contains all elements in the given Collection.
    public boolean containsAll(Collection c);
    // adds all elements in the given Collection
    public boolean addAll(Collection c);
    // removes all elements in the given Collection from self.
    public boolean removeAll(Collection c);
    // removes all elements from the Collection.
    public void clear();

    // and more...
}
```

Data Structures

Each implementation is suited to a different situation:

General-purpose Implementations

Interfaces	Implementations				
	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Queue				LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

Collections

Allows abstract manipulation of collections.

```
public class java.util.Collections {  
  
    public static Object min(List);  
    public static Object max(List);  
    public static void fill(Object, List);  
    public static void reverse(List);  
    // guarantees  $n \log(n)$  time --> optimized merge sort  
    public static void sort(List);  
    public static void shuffle(List);  
    public static int frequency(Object, Collection);  
    public static boolean disjoint(Collection, Collection);  
  
    // and more...  
}
```

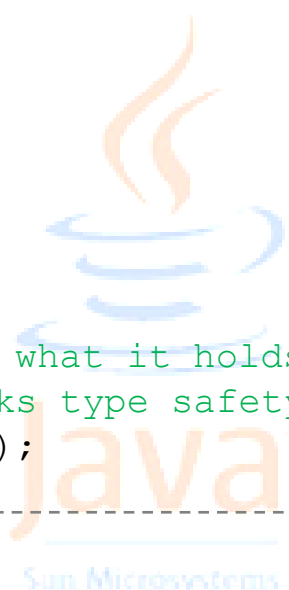
ArrayList

```
List l = new ArrayList();  
  
l.add(2);  
l.add(3);  
l.add(1);  
  
Collections.sort(l);  
  
System.out.print(l.toString());
```



The problem with generic utilities

```
List l = new ArrayList();  
  
l.add(2);  
l.add(3);  
l.add(1);  
  
Collections.sort(l);  
  
// Data structure doesn't know what it holds at compile-time!  
// We have to cast, which breaks type safety  
Integer min = (Integer)l.get(0);
```



GENERICs



<Generics> to the rescue!

```
package java.util;

public interface List {

    public boolean add(Object o);
    public Object get();
    public int size(); // number of objects

    // adds all elements in the given Collection
    public boolean addAll(Collection c);
    // returns true if it contains all elements in the given Collection.
    public boolean containsAll(Collection c);
    // removes all elements in the given Collection from self.
    public boolean removeAll(Collection c);

    // removes all elements from the Collection.
    public void clear();

    // and more...
}
```

<Generics> to the rescue!

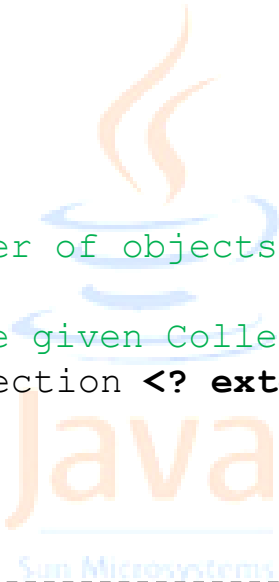
```
package java.util;

public interface List<E> {

    public boolean add(E o);
    public E get();
    public int size(); // number of objects

    // adds all elements in the given Collection
    public boolean addAll(Collection <? extends E> c);

    // and more...
}
```



```
public interface Map<K,V> {
    public V put(K key, V value);
    public V get(K key);
    public void putAll(Map<? extends K,? extends V>);
}
```

Caveats...

```
List<Integer> l = new ArrayList<Integer>();
```

```
l.add(1);
```

```
l.add(2);
```

```
// what if this were possible?
```

```
List<Number> m = l;
```

```
// this is perfectly valid
```

```
m.clear();
```

```
m.add(Math.PI);
```

```
// uh oh!
```

```
Integer i = l.get(0);
```



Wildcards

```
List<?> ints = new ArrayList<Integer>()

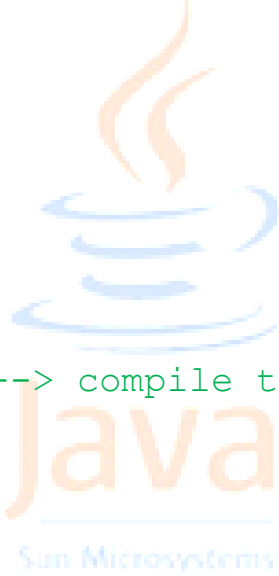
ints.add(0);
ints.add(2);

// valid
List<?> list = ints;
int size = list.size()

// CAN'T DO THIS! Unkown type --> compile time error
list.add(1);

// bounded wildcards
public <E> getRandomElement(List<? extends E> list){
    return list.get((int)(Math.random()*list.size()));
}

public void insertElementRandomly(T element, List<? super T> list) {
    list.add(element, (int)(Math.random()*list.size()));
}
```



Type erasure

```
public class MyClass<E> {  
    public static void myMethod(Object item) {  
        if (item instanceof E) {  
            //Compiler error  
        }  
        E item2 = new E();           //Compiler error  
        E[] iArray = new E[10];      //Compiler error  
        E obj = (E)new Object();     //Unchecked cast warning  
    }  
  
    public static void discoverClass(E item){  
        item.getClass();             // every Object can do this  
    }  
}
```

ENUM



Enums

```
public class Day {
    public static final int
        SUNDAY = 1,
        MONDAY = 2,
        TUESDAY = 3,
        WEDNESDAY = 4,
        THURSDAY = 5,
        FRIDAY = 6,
        SATURDAY = 7;

    private int day;
    public Day(int day){this.day = day; }

    public boolean isWeekend(){
        switch (day) {
            case MONDAY: case TUESDAY: case WEDNESDAY:
            case THURSDAY: case FRIDAY:
                return true;
            case SATURDAY: case SUNDAY: default:
                return true;
        }
    }
}
```

Enums

```
enum Day {
    SUNDAY(true), MONDAY(false), TUESDAY(false), WEDNESDAY(false),
    THURSDAY(false), FRIDAY(false), SATURDAY(true);

    public final boolean isWeekend;
    // constructor
    Day(boolean isWeekend) { this.isWeekend = isWeekend; }
}

Day d = Day.SUNDAY;
boolean b = d.isWeekend;

switch(d) {
    case SUNDAY: //...
    case MONDAY: //...
}

// if it's just to keep track of state
enum ThreadState { STOPPED, RUNNING, WAITING }
```

JAR files



```
C:\$> javac tvald\applet\AnimatedApplet.java
C:\$> javac tvald\intro2java\examples\day4\DemoApplet.java

C:\$> jar cf DemoApplet.jar tvald\applet\AnimatedApplet.class tvald\int
ro2java\examples\day4\DemoApplet*

C:\$> dir
. .. tvald DemoApplet.jar
```

To create a JAR file: `jar cf jar-file input-file(s)`

To view contents: `jar tf jar-file`

To extract contents: `jar xf jar-file`

To run a JAR app: `java -jar app.jar`

Applets from JARs: `<applet code="foo.class" archive="foo.jar"></applet>`

Thanks for coming!

