From C to Java

Hsuan-Tien Lin

Deptartment of CSIE, NTU

OOP Class, March 1-2, 2010

From HelloWorld.c to HelloWorld.java

```c
/* HelloWorld.c */
#include <stdio.h>
int main(){
  printf("Hello World\n");
  return 0;
}
```

```java
/* HelloWorld.java */
import java.lang.*;
public class HelloWorld{
  /** The comment that
   *  will show up in the doc
   */
  public static void main(String[] argv){
    System.out.println("Hello World");//another cmt
  }
}
```

Your Work Cycle

1. edit your Java source file(s)
2. compile
   - `javac HelloWorld.java`
   - output: HelloWorld.class
3. execute
   - `java HelloWorld`
4. generate document
   - `javadoc -d doc/ HelloWorld.java`

What You Can and Cannot Do in Java (1/5)

```c
/* C2Java_1.c */
int main(int argc, char*
    argv[]){

  int i = 3, j = 5, k = 7;
  char c = 'b';
  double d = 3.3;
  i += j; j = k * i;
  k = (j++ % 2) - c;
  i = d * 2;
  c = 'a' + i;
}
```

```java
/* C2Java_1.java */
public class C2Java_1{
  public static void
      main(String[] argv){
    int i = 3, j = 5, k = 7;
    char c = 'b';
    double d = 3.3;
    i += j; j = k * i;
    k = (j++ % 2) - c;
    i = (d * 2);
    c = ('a' + i);
  }
}
```

(int)

(char)

Arithematic: Yes! Automatic Coercion: More Restricted.

Automatic Coercion in Java (Numeric) Types

char(2)

$$\texttt{byte}(1) \rightarrow \texttt{short}(2) \rightarrow \texttt{int}(4) \rightarrow \texttt{long}(8) \rightarrow \texttt{float}(4) \rightarrow \texttt{double}(8)$$

What You Can and Cannot Do in Java (2/5)

```c
/* C2Java_2.c */
int main(int argc,
         char* argv[]){
  int choice = 3,
      people = 4;
  if (choice){
    printf(
      "choice not zero\n"
    );
    if (people >= 5)
      printf(
        "too many people\n"
      );
  }
  else
    printf("%d\n", choice);
  //also: switch/case
}
```

```java
/* C2Java_2.java */
public class C2Java_2{
  public static void
      main(String[] argv){
    int choice = 3,
        people = 4;
    if (choice){
      System.out.println(
        "choice not zero");
      if (people >= 5)
        System.out.println(
          "too many people");
    }
    else
      System.out.println(
        choice);
    //also: switch/case
  }
}
```

(choice != 0)

Conditions: Yes! Nonzero as true (Zero as false): No.

Numerical NOT Boolean

$$\texttt{short} \not\to \texttt{boolean}$$
$$\texttt{int} \not\to \texttt{boolean}$$
$$\texttt{long} \not\to \texttt{boolean}$$
$$\texttt{float} \not\to \texttt{boolean}$$
$$\texttt{double} \not\to \texttt{boolean}$$
$$\texttt{boolean} \not\to \texttt{short}$$
$$\texttt{boolean} \not\to \texttt{int}$$
$$\texttt{boolean} \not\to \texttt{long}$$
$$\texttt{boolean} \not\to \texttt{float}$$
$$\texttt{boolean} \not\to \texttt{double}$$

What You Can and Cannot Do in Java (3/5)

```c
/* C2Java_3.c */
int main(int argc, char*
    argv[]){
    int i = 5, j = 3;
    int sum = 0;
    for(; i ; i--){
        //also: do, while, ...
        for(; j ; j--){
            sum += (i+j);
            if (i == j
                && j == 1)
                goto out;
        }
    }
 out: return 0;
}
```

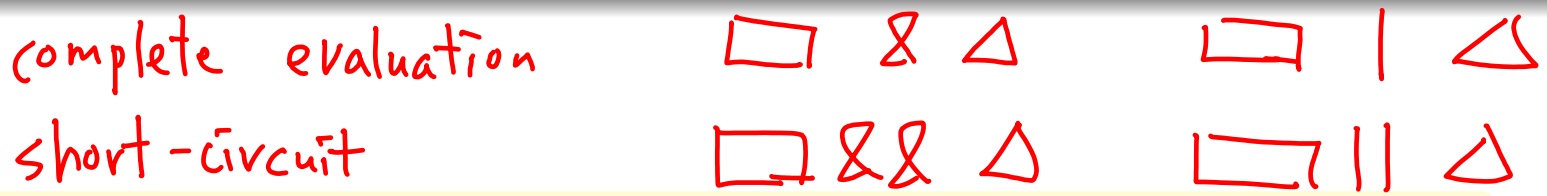
```java
/* C2Java_3.java */
public class C2Java_3{
    public static void
        main(String[] argv){
        int i = 5, j = 3;
        int sum = 0;
myloop: for(; i ; i--){
            //also: do, while,
            ...
            for(; j ; j--){
                sum += (i+j);
                if (i == j
                    && j == 1)
                    goto out;
                    break myloop;
            }
        }
        out: return;
    }
}
```

i != 0

j != 0

Loops: Yes! `goto`: No; `break/continue` with labels: Yes!

goto/break/continue



modern programming philosophy:

- `break/continue` not necessary, but helpful
  —use when making your code easier to maintain
  —especially `continue`
- `goto` usually not necessary (and hence not in Java now)

```
1  do{
2    A;
3    if (B) break;
4    C;
5  }while(D);
```

```
1  do{
2    A;
3    boolean b = B;
4    if (!b) C;
5  }while(!b && D);
```

short circuit, and thus okay

What You Can and Cannot Do in Java (4/5)

```c
/* C2Java_4.c */
int main(int argc, char*
    argv[]){
  char* s = "abc";
  char* t = "123";
  char s2[7];
  printf("%c\n", s[1]);
  printf("%c\n", s[3]);
  strcpy(s2, s);
  strcat(s2, t);
  printf("%s\n", s2);
}
```

```java
/* C2Java_4.java */
public class C2Java_4{
  public static void
      main(String[] argv){
    String s = "abc";
    System.out.println(s[1]);
    System.out.println(s[3]);
    String s2 = s + "123";
    System.out.println(s2);
  }
}
```

hahaha, should use s.charAt(1)

ohohoh, no char at 3

String “Plus”: Yes!

String as Pointer to 0-terminated Char Array: Not really.

Java String

String is a non-primitive, but specially-handled type in Java

`char` is a primitive type, like `short` (2)

What You Can and Cannot Do in Java (5/5)

```c
/* C2Java_5.c */
int main(int argc, char*
    argv[]){
  int a[3] = {1, 2, 3};
  int* b = a;
  int c[] = {7, 8, 9};
  printf("%d\n", c[2] - c[0]);
}
```

```java
/* C2Java_5.java */
public class C2Java_5{
  public static void
      main(String[] argv){
    int a[3] = {1, 2, 3};
    int* b[] = a;
    int c[] = {7, 8, 9};
    System.out.println(
      c[2] - c[0] + c.length
    );
  }
}
```

Array Access: Yes!
Array Construction: Somewhat Different.
Pointers: No More (Yeah?!)

Java Array

Array is another specially-handled type in Java (more in Chapter 6)