C vs Java: A quick overview

Chronology of Computing Ideas

- 1957-66: Fortran I- 66 etc.
- 1972: C Dennis Ritchie: for systems programs
- 1979: C++ Bjarne Stroustrup
- 1994: Java
  - 1996: JDK 1.0 with AWT (Abstract Window Toolkit)
  - 1998: J2SE 1.2 with swing
  - 2004: J2SE 5.0 (Java 2 Platform, St. Ed.) "1.5"
  - 2006: Java SE 6
  - Latest 1.6.0_16 (a.k.a Version 6 Update 16)

C& Java

- **Java is more advanced**
  - **Object oriented**
  - **Graphical user interface: awt, swing**
  - **Many packages and instructions**
- **Java is safer**
  - **Strongly typed, Checks for errors**
  - **No pointer operations**
- **C is very flexible**
  - **Function oriented**
  - **Still used in some applications**
  - **Closer to hardware**

Primitive data types

size in bytes

type	C (typical)	Java (standard)
byte	-	1
char	1 (usually ASCII)	2 (unicode)
short (int)	2	2
int	4 (also long)	4
long long	8	8
float	4	4
double	8	8
boolean	- (0 F, non-0 T)	*

No unsigned numbers in Java
Size in C: char≤short ≤int ≤long
int in C: natural word size

Casting

casting

- C: anything goes
- Java: checked exception at run-time or compile-time

Promotions

- Generally similar, automatic

demotions

- C: automatic, but might lose precision
- Java: must explicitly cast, e.g., to convert from long to int

Strings

- C: null terminated character array
- Java: Strings are objects
  - methods for strings

Pointers

- C: allows pointers to memory locations
  - Allows *, &, + operations
- Java: references

Portability

- C data types depend on machine architecture
  - Java: fixed
- C binary code is architecture specific
  - Java byte code is hardware independent
  - Recompile for each architecture

Hello world

```
#include<stdio.h>
int main(void) {
  printf("Hello\n");
  return 0;
}
```

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

Arrays and allocation

- **Declaration**

  int *a = malloc(N * sizeof(*a)); int[] a = new int[N];
- **allocating memory**

  malloc new
- **de-allocating memory**

  free automatic garbage collection
- **buffer overflow**

  unpredicatable program checked run-time error exception

Compilation & execution

Compilation

- gcc hello.c creates machine code
- javac Hello.java creates bytecode

joint compilation

- gcc main.c helper1.c helper2.c
- javac Main.java - any dependent files are automatically re-compiled if needed

execution

- a.out loads and executes program
- java Hello interprets byte code

Output printing

Printing to standard output
- printf("sum = %d", x);
- System.out.println("sum = " + x);

Formatted printing
- printf("avg = %3.2f", avg);
- System.out.printf("avg = %3.2f", avg)

Reading from stdin
- scanf("%d", &x);
- Java library support,

Objects/data

Data structures:
- C: struct data structures
- Java: Classes and objects

Polymorphism
- C: union
- Java: inheritance

Comments

- **C: /* */**
- **Java: /* */ or //**