Introduction

* A **data type** defines a collection of data objects and a set of predefined operations on those objects
* A **descriptor** is the collection of the attributes of a variable
* An **object** represents an instance of a user-defined (abstract data) type
* One design issue for all data types: What operations are defined and how are they specified?

Integer

* Almost always an exact reflection of the hardware so the mapping is trivial
* There may be as many as eight different integer types in a language
* **Java’s Ints:** byte (1 byte), short (2 bytes), int (4 bytes), long (8 bytes)

Floating Point

* Model real numbers, but only as approximations
* How is 0.1 modeled in binary?
  + 32 Bit IEEE Format
    - 0 01111011 10110011 0011001
    - Sign, Exponent, 23 bits => Adds to 32 bits
* Languages for scientific use support at least two floating-point types (e.g., float and double; sometimes more)
* Usually exactly like the hardware, but not always

Decimal

* Storing the decimal as it is (not using IEEE Format)
* For business applications (money)
  + Essential to COBOL
  + C# offers a decimal data type
* Store a fixed number of decimal digits, in coded from (BCD)
* How is 0.1 modeled in binary?
  + BCD: 4 bits to represent (can store 0-9)
    - 0: 0000
    - 1: 0001
    - 0000.0001
* Advantage: Accuracy
* Disadvantage: Limited range, uses a lot of memory

Boolean

* Simplest of all
* Range of values: True or False
* Could be implemented as bits because only 2 values, but often is as bytes
  + Advantage for this is readability

Character

* Stored as numeric codings
* Most commonly used coding: ASCII
  + Could store 128, then upped to 256, then upped to 65536
* An alternative, 16-bit coding: Unicode (UCS-2)
  + Includes characters from most natural languages
  + Originally used in Java
  + C# and JavaScript also support Unicode

Character String Types

* Values are sequences of characters
* Design Issues:
  + Is it a primitive type or just a special array?
  + Should the length of strings be static or dynamic?
* Primitivity in certain languages
  + C and C++
    - Not primitive
    - Use char arrays and a library of functions that provide operations
  + SNOBOL4 (a string manipulation language)
    - Primitive
    - Many operations, including elaborate pattern matching
  + Fortran and Python
    - Primitive type with assignment and several operations
  + Java
    - Primitive via the Stringclass
  + Perl, JavaScript, Ruby, and PHP
    - Provide built-in pattern matching using regular expressions
* Length Options
  + *Static*: COBOL, Java’s String class
  + *Limited Dynamic Length*: C and C++
    - In these languages, a special character is used to indicate the end of a string’s characters, rather than maintaining the length
  + *Dynamic* (no maximum): SNOBOL4, Perl, JavaScript
* Type Evaluation
  + Aid to writability
  + As a primitive type with static length, they are inexpensive to provide – why not have them?
  + Dynamic length is nice
* Implementation
  + Static length: compile-time descriptor
  + Limited dynamic length: May need a run-time descriptor

Enumeration Types

* All possible values, which are named constants, are provided in the definition
* C# Example
  + **enum** days {mon, tue, wed, thu, fri, sat, sun};
* Design Issues
  + Is an enumeration constant allowed to appear in more than one type definition, and if so, how is the type of an occurrence of that constant checked?
  + Are enumeration values coerced to integer?
  + Any other type coerced to an enumeration type?
* Aid to readability, no need to code a color as a number
* Aid to reliability, compiler can check many things

Array

* An **array**  is a **homogeneous** aggregate of data elements in which an individual element is identified by its position in the aggregate, relative to the first element.