182. Data Structures and Program Design (3)

- Prerequisite: Comp. 110/110L.
- Introduction to data structures and the algorithms that use them.
- Review of composite data types such as
  - arrays,
  - records,
  - strings, and
  - sets.
182. Data Structures and Program Design (3)

- The role of the abstract data type in program design.
  - Definition,
  - implementation, and
  - application of data structures such as
    - stacks,
    - queues,
    - linked lists,
    - trees, and graphs.
182. Data Structures and Program Design (3)

- Recursion.
- Use of time complexity expressions in evaluating algorithms.
- Comparative study of sorting and searching algorithms.
course 110/L Review

• Introduce Java
  – language
  – libraries
  – platform
• Discuss object oriented programming
• Gain experience through labs
Topics of Java Language

- Basics
- Language
- Arrays
- Class
- References
- Constructors
- Static and Final
- String and StringBuffer
- Inheritance
- Dynamic Binding
- Packages
- Protection
- Object
- Exceptions
- Interfaces
- Collections
- Nested Classes
- I/O
- Threads
- Synchronization
- Swing
- Applets
- Networking
Topics of Java Language

- Basics
  - What is Java?
  - Simple application: Hello World

- Java has:
  - Compiler/language
  - Library
  - Virtual machine

- Java Language characteristics
  - simple as C++
  - object oriented
  - interpreted
  - robust
  - secure
  - architecture-neutral
  - portable
  - multithreaded
  - dynamic
Topics of Java Language

- **libraries**
  - i/o
  - strings
  - collections
  - applets
  - networking
  - threads
  - windowing

- **Java Virtual Machine**
  Java code both compiled and interpreted
  - source compiled into bytecodes
  - bytecodes interpreted by virtual machine (VM)

**VM and API**
- hide system specific details
- make "write once, run anywhere" possible
- called platform
program organization

• All Java code organized into classes
  – class represents concept in application domain
  – Ex:
    ```java
    class A
    {
    ...
    }
    ```
program organization

the main() method

• main() is the entry point for Java applications
– arguments passed as array of Strings
– definition must be inside a class

```java
class HelloWorld {
    {
        public static void main(String[] args) {
            
            
        }
    }
}
```
program organization

example

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

source files

• Source files contain classes
  – filename must be same as class
  – with .java extension
• Source files can contain several classes
  – they are in a sequence, at most one class designated "public"
  – filename must be same as public class
**object files**

Object files created by compiler from source file
  – one for each class
  – have .class extension
  – contain bytecodes
java compiler

• JDK (Java Development Kit) comes with compiler
  – javac.exe

• Usage
  – pass name of source file
  – case sensitivity determined by file system

  C: > javac Filename.java

  – if source is stored in Unicode

  C: > javac -encoding unicode Filename.java
java interpreter (VM)

• JDK comes with interpreter
  – java.exe
• Usage
  – pass name of object w/o suffix .class
  – object name is case sensitive even on case insensitive file system

C:\> java ObjectName
java language

- Language constructs
  - comments
  - primitive types
  - variables
  - scope
  - operators
  - expressions
  - control constructs
comments

// comment
    the comment will stop at the end of the current line

/**< comment */
    - can span multiple lines,
    - can overwrite // comments
    - cannot be nested

/**< comment */
    - It is used in javadoc

Where to use them in program?
Primitive types

- byte, short, int, long
- float, double
- boolean
- char

Examples:

```java
int age, books;
double gpa;
boolean sexType;
char letterGrade;
```
Variable

- made from Unicode characters
  - letters, digits, underscore, etc.
  - case is significant
  - no operators (+-*/; etc.)
  - cannot start with digit
- Variables can be initialized when defined
  - assign value of appropriate type
  - note that float constants require suffix 'F' or 'f'

Examples:
  int id = 1002;
  double π = 3.14;
  float gpa = 4.0f;
  boolean isMale = true;
  char grade = 'B';
scope

- **block scope:** curly braces define a scope
  - Variables can be defined in a block
  - accessible only within block

- **Class block**
expressions

• Standard mathematical operations available
  – operators + - * /

  Precedence * / higher than + -

  Parentheses can be used to force an association

• Shorthand operators
  – increment ++
  – decrement --
  – assignment +=, -=, *=, /=,
Types conversion

• Int division yields an integer
  – fractional part lost
  – modulus operator % gives remainder
• Mixed type expressions allowed
  – operands automatically converted to common type
  – smaller type converted to larger
  – called promotion
  – result is of the larger type
• A cast is a forced type conversion
  – type name put in parentheses
    ex: f =((float) i)/j; // to have a float division
• Notes:
  – promotion occurs automatically
  – demotion requires explicit cast
Control keywords

- choice:
  - if
  - If <> then <>
  - If <> then <> else <>

- Loops
  - while
  - do
  - for

- break
- continue
- switch
Relational operators

<   less than
<=  less than or equal
>   greater than
>=  greater than or equal
==  equal
!=  not equal
Logical operators used to combine expressions

&& and
|| or
! not
**LOOPS**

**while**

- while loop provides iteration
  - loop condition evaluated before entering

```javascript
while (conditions)
{
}
```

**do**

- do loop executed at least once

```javascript
do
{
} while (conditions);
```
LOOPS

for
  • for loop centralizes control information
    ```
    for (int i = 0 ; i < 5 ; i++)
    {
    }
    ```

break
  • `break` statement used to exit loop

continue
  • `continue` shortcuts loop iteration
    skips rest of body for that iteration
LOOPS

**switch**

- *switch* selects one option from several choices

```java
switch (grade)
{
    case 'A': sum += 4.0; break;
    case 'B': sum += 3.0; break;
    case 'C':
    case 'D': sum += 1.0; break;
    case 'F': sum += 0.0; break;
    default:  // none of above
}
```