3 Identifiers, Keywords & Types

Objectives

- Use comments in a source program
- Distinguish between valid and invalid identifiers
- Recognize Java technology keywords
- List the eight primitive types
- Define literal values for numeric and textual types
- Define the terms *class*, *object*, *member variable*, and *reference variable*
- Create a class definition for a simple class containing primitive member variables
- Declare variables of class type
- Construct an object using new
- Describe default initialization
- Access the member variables of an object using the dot notation

Comments

- Three permissible styles of comment in a Java technology program are :

// comment on one line

/* comment on one

or more lines */

/** documenting comment */

JAVADOC

Javadoc is a tool that parses the declarations and documentation comments in a set of source files and produces a set of HTML pages describing the classes, inner classes, interfaces, constructors, methods, and fields.

```
javadoc -sourcepath $(SRCDIR)
        -overview $(SRCDIR)/overview.html
        -d /java/jdk/build/api
        -use
        -splitIndex
        -windowtitle $(WINDOWTITLE)
        -doctitle $(DOCTITLE)
        -header $(HEADER)
        -bottom $(BOTTOM)
        -group $(GROUPCORE)
        -group $(GROUPEXT)
```


JAVADOC TAGS

Javadoc parses special tags when they are embedded within a Java doc comment. These doc tags enable you to autogenerate a complete, well-formatted API from your source code.

Tag	Introduced in JDK/SDK
@author	1.0
{@docRoot}	1.3
@deprecated	1.0
@exception	1.0
{@link}	1.2
@param	1.0
@return	1.0

Tag	Introduced in JDK/SDK
@see	1.0
@serial	1.2
@serialData	1.2
@serialField	1.2
@since	1.1
@throws	1.2
@version	1.0

Semicolons, Blocks, and White Space

- A statement is one or more lines of code terminated by a semicolon (;) :

```
totals = a + b + c
         + d + e + f ;
```

- A block is a collection of statements bound by opening and closing braces:

```
{
   x = y + 1 ;
   y = x + 1 ;
}
```

Semicolons, Blocks, and White Space

- A *block* can be used in a *class* definition

```
public class Date {
   private int day;
   private int month;
   private int year;
}
```

- Block statements can be nested
- Any amount of *whitespace* is allowed in a Java program


```
while ( i < large ) {
   a = a + i ;
   // nested block
   if ( a == max ) {
      b = b + a ;
      a = 0 ;
   }
   i = i + 1 ;
}
```


Identifiers

- Are names given to a variable, class, or method
- Can start with a letter, underscore(_), or dollar sign($)
- Are case sensitive and have no maximum length
- Examples:

```
identifier
userName
user_name
_sys_var1
$change
```


Java Keywords

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
void	volatile	while		

const ve **goto**, Java'da tanımlı olmasalar da değişken isimleri olarak kullanılamazlar.

Primitive Types

- The Java programming language defines eight primitive types
  - Logical - **boolean**
  - Textual - **char**
  - Integral - **byte**, **short**, **int**, and **long**
  - Floating - **double** and **float**

Logical – boolean

- The boolean data type has two **literals**, `true` and `false`.
- For example, the statement

```
boolean truth = true;
```

declares the variable `truth` as `boolean` type and assigns it a value of `true`.

Textual — char and String

`char`

- Represents a 16-bit Unicode character
- Must have its literal enclosed in single quotes(' ')
- Uses the following notations:

`'a'`	The letter a
`'\t'`	A tab
`'\u????'`	A specific Unicode character (????) is replaced with exactly four hexadecimal digits. For example, '\u03A6' is the Greek letter phi[Φ]

Textual — char and String

String

- Is not a primitive data type; **it is a class**
- Has its literal enclosed in double quotes (" ")

```
"The quick brown fox."
```

- Can be used as follows:

```
String greeting = "Good Morning !!\n";
String err_msg = "Record Not Found !";
```



```
// declares and initializes a char variable
char ch = 'A' ;
// declares two char variables
Char ch1,ch2 ;
// declares two String variables and initializes them
String greeting = "Good Morning !!\n" ;
String errorMessage = "Record Not Found !!\n" ;
// declares two String variables
String str1,str2 ;
```

Note: initial values for str1 and str2 are null. Without initialization

```
System.out.println(str1) ;
```

causes to print null.

Integral — byte, short, int, and long

- Uses three forms - decimal, octal, or hexadecimal

`2`	The decimal value is two.
`077`	The leading zero indicates an octal value.
`0xBAAC`	The leading `0x` indicates a hexadecimal value.

- Has a default `int`
- Defines `long` by using the letter `"L"` or `"l"`

Integral — byte, short, int, and long

Each of the integral data types have the following range:

Integer Length	Name or Type	Range
8 bits	byte	-2^7 to 2^7-1
16bits	short	-2^{15} to $2^{15}-1$
32bits	int	-2^{31} to $2^{31}-1$
64bits	long	-2^{63} to $2^{63}-1$

Floating Point — float, double

- Default is double
- Floating point literal includes either a decimal point or one of the following:
  - `E or e (add exponential value)`
  - `F or f (float)`
  - `D or d (double)`

`3.14`	A simple floating-point value (a `double`)
`6.02E23`	A large floating-point value
`2.718F`	A simple `float` size value
`123.4E+306D`	A large `double` value with redundant D

Floating Point – float, double

Floating-point data types have the following ranges:

Float Length	Name or Type
32 bits	float
64 bits	double


```
public class Assign {
      public static void main(String[] args){
          int x,y ;
          float z = 3.4115 ;
          boolean truth = true ;
          char c ;
          String str ;
          String str1 = "bye"  ;
          c = 'A' ;
          str = "Hi out there!" ;
          x = 6 ;
          y = 1000 ;
      }
}
```



```
y = 3.1415926;  // 3.1415926 is not an int; it
                // requires casting and decimal will be truncated

w = 175,000 ;  // Comma symbol cannot appear

truth = 1 ;  // this is a common mistake
             // made by C/C++ programmers

z = 3.1415926 ;  // Can't fit double into a
                 // float; This requires casting
```


Java Reference Types

- Beyond primitive types all others are of reference types
- A *reference variable* contains a handle to an object.
- Example:

next slide


```
public class MyDate {
        private int day = 1 ;
        private int month = 1 ;
        private int year = 1923 ;
        public MyDate(int day,int month,int year) {
           this.day = day ;
           this.month = month ;
           this.year = year ;
        }
        public void print(){...}
 }
public class TestMyDate {
     public static void main(String[] args){
          MyDate today = new MyDate(22,7,1964) ;
     }
}
```


Constructing and Initializing Objects

- Calling *new ClassName()* to allocate space for the new object results in:
  - Memory allocation: Space for the new object is allocated and instance variables are initialized to their default values,
  - Explicit attribute initialization is performed
  - A constructor is executed
  - Variable assignment is made to reference the object
- Example:

```
MyDate my_birth = new MyDate(11,7,1973) ;
```

Memory Allocation and Layout

▶A declaration allocates storage only for a reference:

```
MyDate my_birth = new MyDate(11,7,1973) ;
```

my_birth	?????

▶Use the new operator to allocate space for MyDate:

```
MyDate my_birth = new MyDate(11,7,1973) ;
```

my_birth	?????
day	0
month	0
year	0

Explicit Attribute Initialization

▶Initialize the attributes:

```
MyDate my_birth = new MyDate(11,7,1973) ;
```

my_birth	?????
day	1
month	1
year	1923

▶The default values are taken from the attribute declaration in the class.

Executing the Constructor

▶Execute the matching constructor:

```
MyDate my_birth = new MyDate(11,7,1973) ;
```

my_birth	?????
day	11
month	7
year	1973

▶In the case of an overloaded constructor, the first constructor may call another.

Assigning a Variable

▶Assign the newly created object to the reference variable:

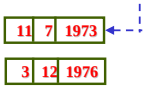
```
MyDate my_birth = new MyDate(11,7,1973) ;
```

my_birth	0x01abcdef
day	11
month	7
year	1973

Assigning Reference Types

Consider the following code fragment:

int x = 7 ;	*x*	7
int y = x ;	*y*	7
MyDate s = new MyDate(11,7,1973) ;	*s*	0x01234567
MyDate t = s ;	*t*	0x01234567
t = new MyDate(3,12,1976) ;	*t*	0x12345678



Pass-by-Value

- The Java programming language only passes arguments by value,
- When an object instance is passed as an argument to a method, the value of the argument is a reference to the object,
- The contents of the object can be changed in the called method, but the object reference is never changed.
- Example:

↳ next slide

```
public class PassTest {
        public static void changeInt(int myValue) {
            myValue = 55 ;
        }
        public static void changeObjectRef(MyDate ref) {
            ref = new MyDate(1,1,2000) ;
        }
        public static void changeObjectAttr(MyDate ref) {
            ref.setDay(4) ;
        }
        public static void main(String[] args){
            MyDate date ;
            int val ;
            val = 11 ;
            changeInt(val) ;
            System.out.println("Int value is: "+val) ;
            date = new MyDate(22,7,1964);
            changeObjectRef(date) ;
            date.print();
            changeObjectAttr(date);
            date.print();
        }
}
```


The this Reference

- Here are a few uses of the this keyword:
  - To reference local attribute and method members within a local method or constructor
  - The keyword `this` distinguishes a local method or constructor variable from an instance variable
  - To pass the current object as a parameter to another method or constructor

1

```
public class Circle {
  public double x, y, r;  // The center and the radius of the circle
  public Circle ( double x, double y, double r ) {
    this.x = x;  this.y = y;   this.r = r;
  }
  public double circumference( ) { return 2 * 3.14159 * r; }
  public double area( ) { return 3.14159 * r*r; }
}
```


2

```
public class Circle {
  public double x, y, r;
  // An instance method.  Returns the bigger of two circles.
  public Circle bigger(Circle c) {
    if (c.r > this.r) return c; else return this;
  }
  // A class method.  Returns the bigger of two circles.
  public static Circle bigger(Circle a, Circle b) {
    if (a.r > b.r) return a; else return b;
  }
  // Other methods omitted here.
}
```


Java Programming Language Coding Conventions

- Packages:

```
package shipping.object ;
```

- Classes

```
class AccountBook ;
```

- Interfaces

```
interface Account
```

- Methods

```
balanceAccount()
```


Java Programming Language Coding Conventions

- Variables:

```
currentCustomer
```

- Constants:

```
HEAD_COUNT
MAXIMUM_SIZE
```

3# Identifiers, Keywords, and Types

- Exercise-1: “Investigating Reference Assignment”
- Exercise-2: “Creating Customer Accounts”

101