hi-tech college + JOHN BRYCE Leading in IT Education a matrix company

Expressions and Flow Control

Variable's Scope

- Local variables and local scope.
- Global variables.

Local Variables

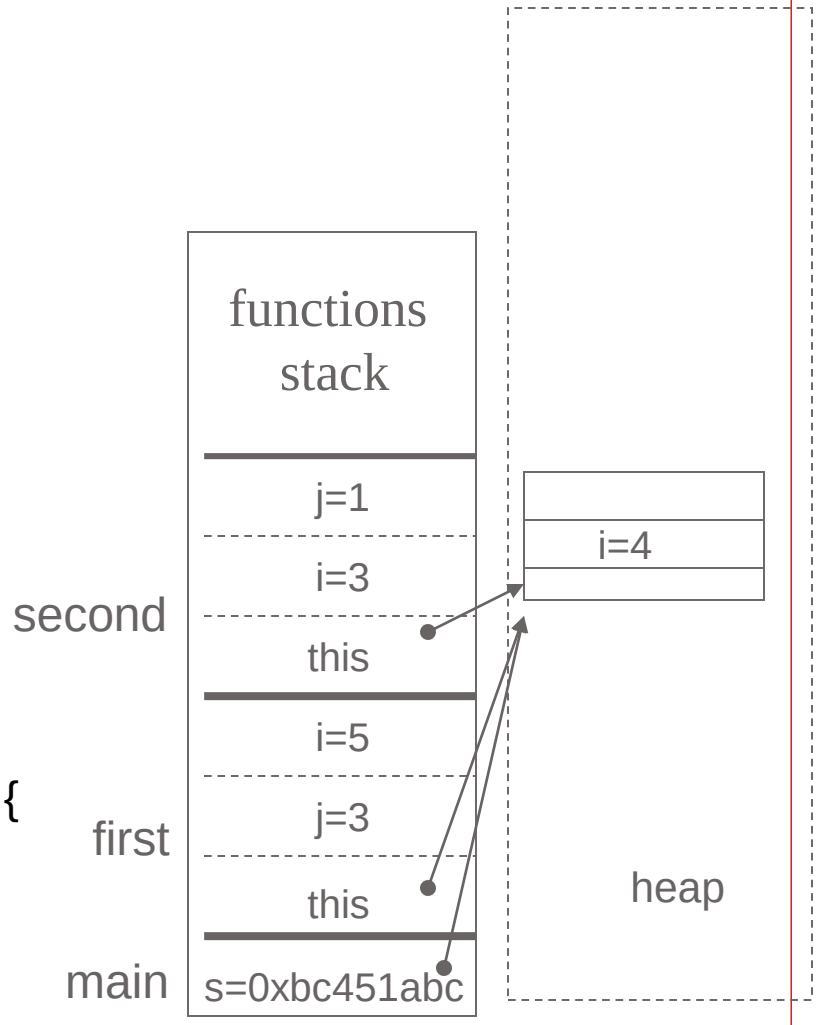
- Are defined inside a method and are called *local*, *automatic*, *temporary,* or *stack* variables.
- Are created when the method is executed and are destroyed when the method is exited.
- Uninitialized automatically. Lack of initialization will result in compile time error.

Classes and Objects Variables

- Are initialized automatically.
- Object variables exist in the scope of an object.
- Class variables are global.

Scope Example

```
public class Scope {
    private int i=9;
    public void first()     {
        int j=3,i;
        i=5;
        this.i=i+j;
        second(j);
    }
    public void second (int i)     {
        int j=1;
        this.i=i+j;
    }
    public static void main(String args[])    {
        Scope s=new Scope();
        s.first();
    }
}
```



Precedence of Operators

Associative	Operators
R to L	++ -- + - ~ ! (data type)
L to R	* / %
L to R	+ -
L to R	<< >> >>>
L to R	< > <= >= instanceof
L to R	== !=
L to R	&
L to R	^
L to R	|
L to R	&&
L to R	||
R to L	?:
R to L	= *= /= %= += -= <<= >>= >>>= &= ^= |=

Logical Operators

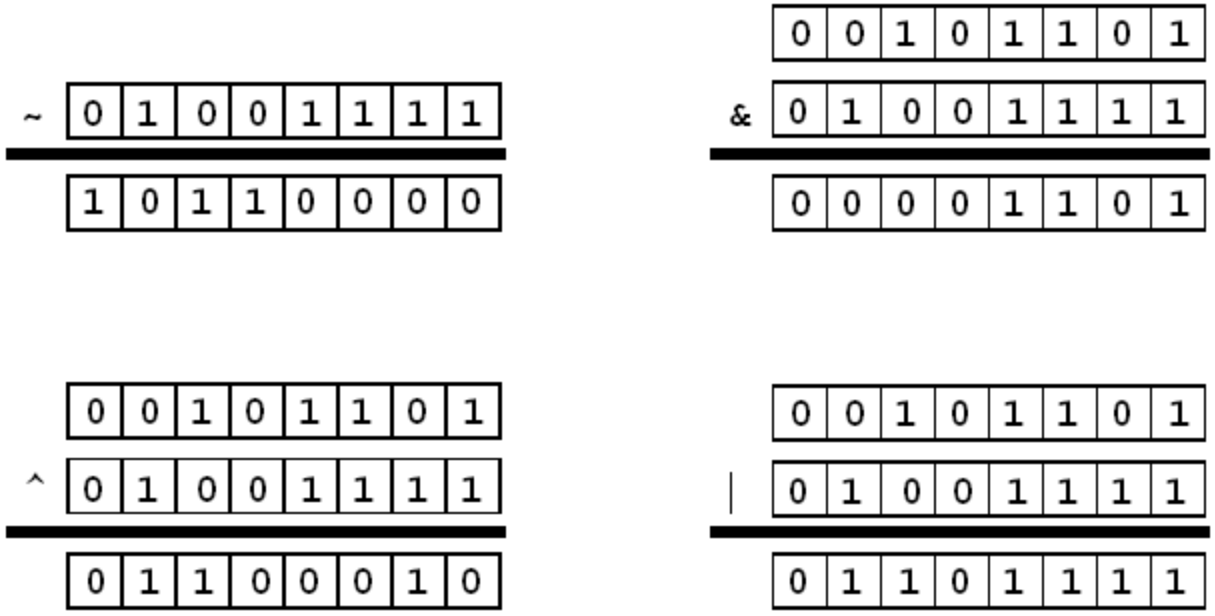
NOT - ! OR - || AND - &&

Bits Operators

OR - | XOR - ^ AND - &

ONE'S COMPLEMENT - ~

Bitwise Logical Operators - Example



Shift Operators

>> - Signed right shift.
>>> - Unsigned right shift.
<< - Left shift.

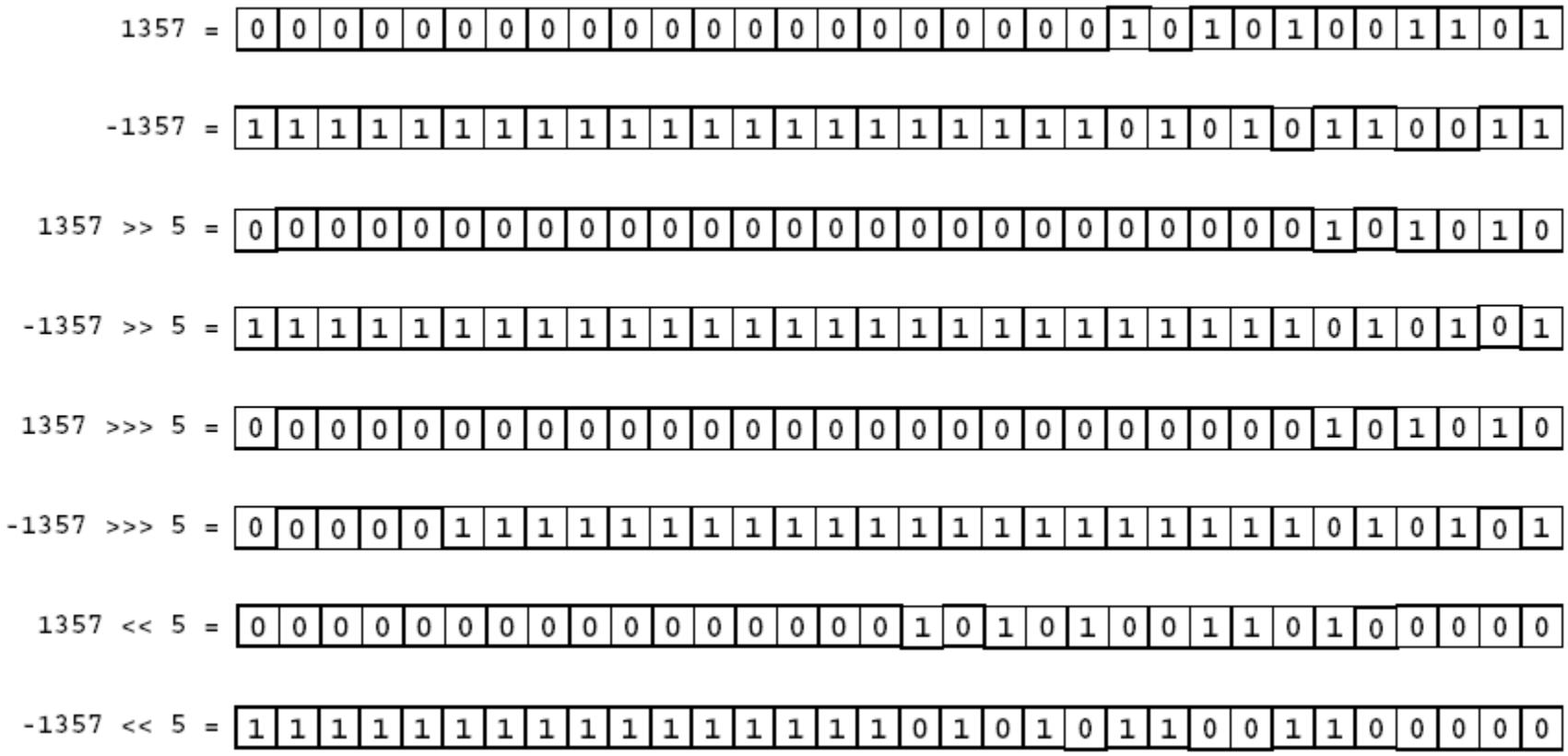
- Syntax:

  num >> no_of_right_shifts
  num >>> no_of_right_shifts
  num << no_of_left_shifts

- Example:

  30>>4 = $30/2^4$
  30<<4 = $30*2^4$
  -30>>>4 = $30/2^4$

Shifts Examples



String Concatenation

- String objects are concatenated using the ‘+’ operator.
- One of the objects must be a string and the other one will be converted automatically to a string.
- An object may be cast to a string using the toString( ) method (will be explained in a later module).

String Concatenation - Example

```
String s = “Hello ”;
String name= “kuky”;
int num = 3;
s = s+”to “+name+” and his “+num+” dogs.”;
System.out.println(s);
        /* “Hello to kuky and his 3 dogs.”
                will be printed.  */
```

Casting

- Explicit cast is required when assigning a larger type value into a smaller type variable.
- A cast will not affect the right side value.
- In a mixed type expression, variables are automatically promoted to a larger type.

Casting Examples

```
int num = 100L ;                    //error
float fnum = 234.78 ;               //error, default value of a floating point
                                    //number is double.
int num = 10;                       //fine.
float fnum = 5.4f;                  //fine.
float fnum = (float) 4.44;          //fine.
double dnum = 7.5f;                 //fine, an assignment from float to double,
                                    //can not result in information lost.
int n = 5;
long l_num=n;                       //fine, an assignment from int to long,
                                    //can not result in information lost.
```

Branching Statements - If

- An if statement:

```
if (boolean expression) {
        statement or block;
}
```

- An if-else statement:

```
if ( boolean expression) {
        statement or block;
}
else {
        statement or block;
}
```

An If-Else Example

```
If (no_of_student > MAX_STUDENTS)
   openAnotherClass( );
else if (no_of_student < MIN_STUDENTS)
   closeClass( );
else
   enterClass( );
```

Branching Statements – Switch

The switch statement syntax:

```
switch (expr1) {
case constant2:
        statements;
        break;
case constant3:
        statements;
        break;
default:
        statements;
        break;
}
```

Branching Statements – Switch

- Cases may be one of the following:

byte

short

char

Int

Example:

```
int x=getXFromUser();
switch(x){
…
```

Loops in Java - The *for* Loop

```
for (init_expr; boolean_test_expr; alter_expr) {
        statement or block;
}
```

Example:

```
for (i=0; i< NO_OF_ITERATIONS; i++)
        System.out.println("Counter is: "+i);
```

Loops in Java-The *while* Loop

- while (boolean_test_expr) {
  statement or block ;
  }

- Example*:*

```
int i=0;
while ( i< NO_OF_ITERATIONS) {
    System.out.println("Counter is: "+i);
    i++;
}
```

Loops in Java - The *do-while* Loop

- do {
  statement or block ; }
  while (boolean_test_expr) ;

- Example*:*

```
int i=0;
do {
    System.out.println("Counter is: "+i);
    i++; }
while ( i< NO_OF_ITERATIONS) ;
```

Loop Flow Control

- break [label];
- continue [label];
- label: loop;

Loop Flow Control

- The ***break*** statement will lead to permanently exit from the loop (“break” the loop).
- The ***continue*** statement will lead to exit the current iteration and to continue the flow of the loop from the beginning of the next iteration.
- The ***label*** statement identifies any valid statement to which the control must be transferred. For a ***break*** statement, a valid ***label*** may be any legal statement. For a ***continue*** statement, a valid ***label*** must identify a loop.

The *break* statement:

```
do {
        statement;
        if ( condition is true) {
                break;
        }
        statement;
} while ( boolean expression);
```

The *continue* statement:

```
do {
        statement;
        if ( condition is true) {
                continue;
        }
        statement;
} while ( boolean expression);
```

Loop Flow Control - The *break* Statement

```
outer: do {
                    statement;
                    do {
                            statement;
                            if ( boolean expression) {
                                    break outer;
                            }
                            statement;
                        } while ( boolean expression);
          statement;
        } while ( boolean expression);
```

Loop Flow Control-The *continue* statement

```
outer: do {
            statement;
            do {
                  statement;
                  if ( boolean expression) {
                        continue outer;
                  }
                  statement;
              } while ( boolean expression);
            statement;
      } while ( boolean expression);
```