Java: Type Conversion, Statements and Loops

Instructor: Nihshanka Debroy
Type Conversion: Type I

• Assignment Conversion

```java
int smallerType = 2;

// 1) ASSIGNMENT CONVERSION (converts the int to a double)
double largerType = smallerType;
System.out.println("Assignment Conversion: "+largerType);
```
Type Conversion: Type II

• Arithmetic Promotion

// 2) ARITHMETIC PROMOTION (answer will be a double)

double big = 100.36;

int small = 2;

double answer = big / small;

System.out.println("\nArithmetic Promotion:"+answer);
Type Conversion: Type III

- Casting

```java
// 3) CASTING

double d = 100.36;
int i = (int)d;

System.out.println("\nCasting:");
System.out.println("Integer Value = "+i);
System.out.println("Double Value = "+d);
```
Automatic type conversion

- Mixed type expressions are converted to higher compatible types
- If all operands are of type int then result is type int
- If any operand is of type double then result is of type double
- Cannot convert to a lower type
- Conversion may result in loss of precision

Example:
Convert Fahrenheit to Celsius

double F=41.0;
double C=(F-32.0)*(5/9);

Question: What is the value of C?

a) 5
b) 0.0
c) 9.0
d) 5.0
e) 9
8 primitive data types: mnemonic aid

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

Be
Careful,
Bears
Shouldn't
Ingest
Large
Furry
Dogs
Control structures

- Determine flow of execution of a program’s instructions
  - Sequential execution
    • instructions follow one another in a logical progression
  - Selective execution
    • provides a choice depending upon whether a logical expression is true or false
  - Repetitive execution
    • the same sequence of instructions is to be repeated a number of times
- We can construct any algorithm using combinations of control structures
Sequential execution

- statement 1
- statement 2
  ...
- statement $n$
Selective Execution
if selection

- expression
  - true → statements
  - false
    - return to expression

Selective execution: if-else

- Executes one set of statements when logical expression is true and different set of statements when expression is false.
- Used to select between two cases.
- Multiple statements are enclosed in curly braces.

```plaintext
if (logical-expression )
{
    ... statements1 ...
}
else
{
    ... statements2 ...
}
```
if-else selection

false

expression

ture

statements1

statements2
If-else examples

```java
int value1=4;
int value2=5;

if(value1 > value2)
    System.out.println("value1 is larger");
else
    System.out.println("value2 is larger");

if(value1 > value2) {
    System.out.println("Which value is larger?");
    System.out.println("value1 is larger.");
}
else {
    System.out.println("Actually...");
    System.out.println("value2 is larger.");
}
```
Nested if-else

if (num1 < num2)  // pair 1
  if (num1 < num3)  // pair 2
    min = num1;
  else  // pair 2
    min = num3;
else  // pair 1
  if (num2 < num3)  // pair 3
    min = num2;
  else  // pair 3
    min = num3;

What will happen if:

- num1=4
- num2=2
- num3=3
Repetitive Execution: Loops
Anatomy of a for loop

```java
int i;
for(i=1;i<=10;i++)
{
    System.out.println("Hello World");
}
```

This can be interpreted as:

Start by setting $i=1$.
As long as ($i<=10$), keep doing this:

1. $i=1$ (initialization step)
2. $i<=10$ (condition that we check each time)
3. $i++$ (increase $i$ by 1), is the update/increment step
4. More than one statement in loop: use curly braces
for-loop example

- Q: If \( a=1 \), \( b=3 \), and \( x=7 \), what is the value of \( x \) when the loop terminates?

```java
for (k=a; k<=b; k++)
{
    x=x-k;
}
```

- A: \( x=1 \)

- First iteration (\( k=1, x=7 \))
  - test: \( 1<=3 \) is true
  - execute: \( x=7-1=6 \)
  - update: \( k=2 \)
- Second iteration (\( k=2, x=6 \))
  - test: \( 2<=3 \) is true
  - execute: \( x=6-2=4 \)
  - update: \( k=3 \)
- Third iteration (\( k=3, x=4 \))
  - test: \( 3<=3 \) is true
  - execute: \( x=4-3=1 \)
  - update: \( k=4 \)
- Fourth iteration (\( k=4, x=1 \))
  - test: \( 4<=3 \) is false
  - exit loop
Another for-loop example

- Q: If \( a=2, \ b=4, \ x=1, \) and \( y=9, \) what are the values of \( x \) and \( y \) when the loop terminates?

  ```
  \text{for}(k=a; \ k<b; \ k++)
  \begin{align*}
  &x=x+k; \\
  &y=y-x;
  \end{align*}
  ```

- A: \( x=6, \ y=0 \)

- First iteration (\( k=2, \ x=1, \ y=9 \))
  - test: 2<4 is true
  - execute: \( x=1+2=3 \)
    \( y=9-3=6 \)
  - update: \( k=3 \)

- Second iteration (\( k=3, \ x=3, \ y=6 \))
  - test: 3<4 is true
  - execute: \( x=3+3=6 \)
    \( y=6-6=0 \)
  - update: \( k=4 \)

- Third iteration (\( k=4, \ x=6, \ y=0 \))
  - test: 4<3 is false
  - exit loop
The Factorial Example with a for loop

public class Math {
    public static void main(String args[]) {
        int iterations; // number of times we'll run through the loop
        int answer = 1; // holds the value that we're calculating
        // prompting the user for input //
        System.out.print("Enter the number you want the factorial of, here->");
        iterations = Keyboard.readInt();
        /* calculating the answer */
        for(int i = 1; i <= iterations; i++)
            answer = i * answer;
        // printing out the answer
        System.out.println("The answer is: "+answer);
    }
}
while loop repetition

- Expression

  True -> Statements

  False

Anatomy of a while loop

```java
int i = 1;
while (i <= 10)
{
    System.out.println("Hello World");
    i = i + 1;
}
```

This can be interpreted as:

Start by setting \(i=1\).

As long as \(i \leq 10\), keep doing this:

- \(i=1\) (initialization step)
- \(i \leq 10\) (condition that we check each time)
- \(i=i+1\) (increase i by 1), is the update/increment step
while-loop example

• Q: What is the value of \( p \) when the loop terminates?

```
p=0;
t=1;
n=10;
while(n>t)
{
    p=p+n*t;
    t=t+4;
}
• A: p=150
```

• First iteration (\( t=1, p=0 \))
  – test: \( 10>1 \) is true
  – execute: \( p=0+10*1=10 \)
    \( t=1+4=5 \)

• Second iteration (\( t=5, p=10 \))
  – test: \( 10>5 \) is true
  – execute: \( p=10+10*5=60 \)
    \( t= 5+4=9 \)

• Third iteration (\( t=9, p=60 \))
  – test: \( 10>9 \) is true
  – execute: \( p=60+10*9=150 \)
    \( t= 9+4=13 \)

• Fourth iteration (\( t=13, p=150 \))
  – test: \( 10>13 \) is false
  – exit loop
Another while-loop example

Q: What are the values of z, p, and n after executing the following statements?

```java
n=-1;
z= 0;
p= 1;
while(p<=10)
{
    z=n*z*p;
    p=p+4;
    n=n-3;
}
```

A: z=0, p=13, n=-10

First iteration (n=-1, z=0, p=1)
- test: 1<=10 is true
- execute: z=-1*0*1=0
  p= 1+4=5
  n=-1-3=-4

Second iteration (n=-4, z=0, p=5)
- test: 5<=10 is true
- execute: z=-4*0*5=0
  p= 5+4=9
  n=-4-3=-7

Third iteration (n=-7, z=0, p=9)
- test: 9<=10 is true
- execute: z=-7*0*9=0
  p= 9+4=13
  n=-7-3=-10

Fourth iteration: exit loop
**do while loop**

i = 1;

do {
    System.out.print("This is a do while loop");
    i = i + 1;
} while (i < 10);

This can be interpreted as:

Start by setting i = 1.

Keep doing this:

{  
    print "This is a do while loop"
    Increase i by 1
} As long as i < 10

**Important:** do while loop gets executed at least once, as opposed to a while loop (because in a do while loop, the condition gets checked only after the first iteration)
For-loop or while-loop?

- When to use a for-loop:
  - always for counting!
  - you know how many times to execute the loop
- When to use a while-loop:
  - number of repetitions needed is unknown

```plaintext
for(k=a; k<b; k++)
{
    x=x+k;
}
```

A for loop can always be written as a while loop

```plaintext
k=a;
while(k<b)
{
    x=x+k;
    k=k+1;
}
```
Infinite Loops

What's wrong with this piece of code?

```java
int i = 1;
while (i <= 10) {
    System.out.println("Hello World");
}
```