CSCI 1304: Programming I

Chapter 2 Basic Computation

Part 2 Identifiers, Assignments, Expressions, and Constants

Dr. Nour Almadhoun Alserr

Week 4

1/10/2023

Objectives

- Write Java statements to **declare variables** and define named constants.
- Write **assignment statements** and **expressions** containing variables and constants.

Java identifiers

- **Identifier**: the name of something in a Java program, such as a **variable**, **class**, or **method**.
- It **must not** start with a **digit** and may contain:
  - Letters,
  - Digits 0 through 9,
  - The underscore character (_)
  - The symbol $ is also allowed (reserved)

- Since Java is *case sensitive*, `iug`, `Iug`, and `IUG` are different identifiers.

Java identifiers

```
public class Test
{
    public static void main(String[] args)
    {
        int a = 20;
    }
}
```

Java identifiers

```
public class Test
{
    public static void main(String[] args)
    {
        int a = 20;
    }
}
```

Java identifiers

```
public class EggBasket
{
    public static void main (String [] args)
    {
        int numberOfBaskets, eggsPerBasket, totalEggs;
        numberOfBaskets = 10;
        eggsPerBasket = 6;
        totalEggs = numberOfBaskets * eggsPerBasket;
        System.out.println ("If you have");
        System.out.println (eggsPerBasket + " eggs per basket and");
        System.out.println (numberOfBaskets + " baskets, then");
        System.out.println ("the total number of eggs is " + totalEggs);
    }
}
```

Java identifiers

- **What are the valid identifiers ?!**

miles

public

int

a++

$4

Four_Students$2021

Test

apps

4#R

x

class

radius

--a

Four Students$2021

#44

apps

y

Keywords or Reserved Words

- Words such as `if` are called ***keywords*** or ***reserved words*** and have **special, predefined** meanings.
  - Cannot be used as identifiers.

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

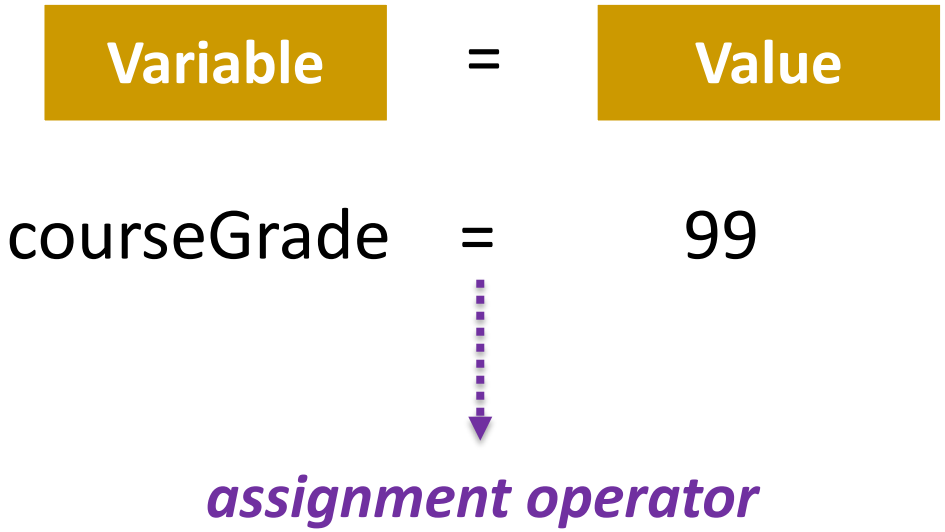
Where to Declare Variables

- Just **before using it**, or
- At the **beginning** of the section of your program (**main method**) that is enclosed in { }.

```
public static void main(String[] args)
{ /* declare variables here */
    . . .
}
```

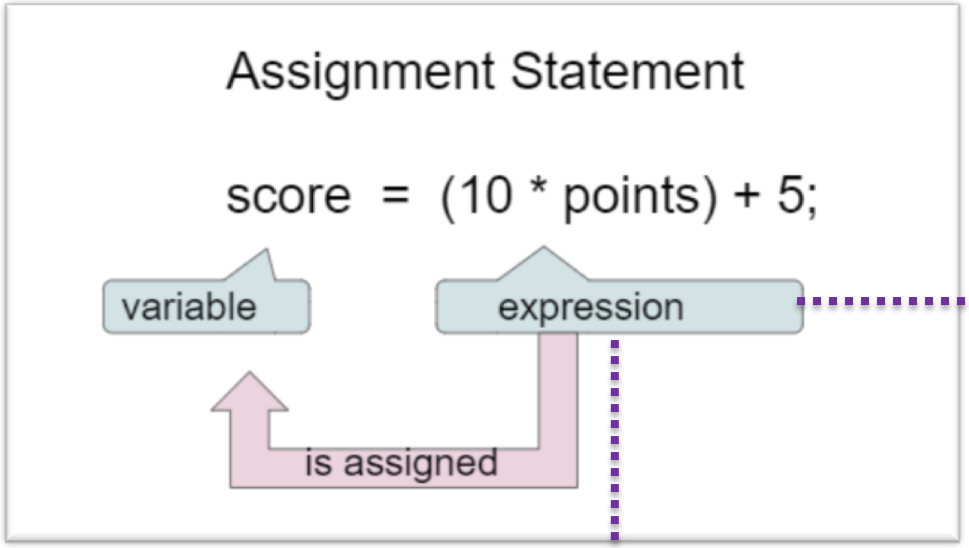
Assignment Statements

- An assignment statement is used to **assign** a value to a variable.



Assignment Statements

Syntax



- Another variable.
- *Literal* or *constant (e.g., number).*
- Variables and literals with *operators (+,-).*

Assignment Examples

```
amount = 3.99;
firstInitial = 'W';
FinalGrade = midterm + FinalExam;
score = numberOfCards + handicap;
eggsPerBasket = eggsPerBasket - 2;
byte = 8;
```

Assignment Statements

```
1
2  public class VariableAssignment
3  {
4    public static void main(String[] args)
5    {
6      int x = 3;
7      int y = 2;
8      System.out.println(x);
9      System.out.println(y);
10     x = y;
11     System.out.println(x);
12     System.out.println(y);
13     y = 5;
14     System.out.println(x);
15     System.out.println(y);
16   }
17 }
18
```

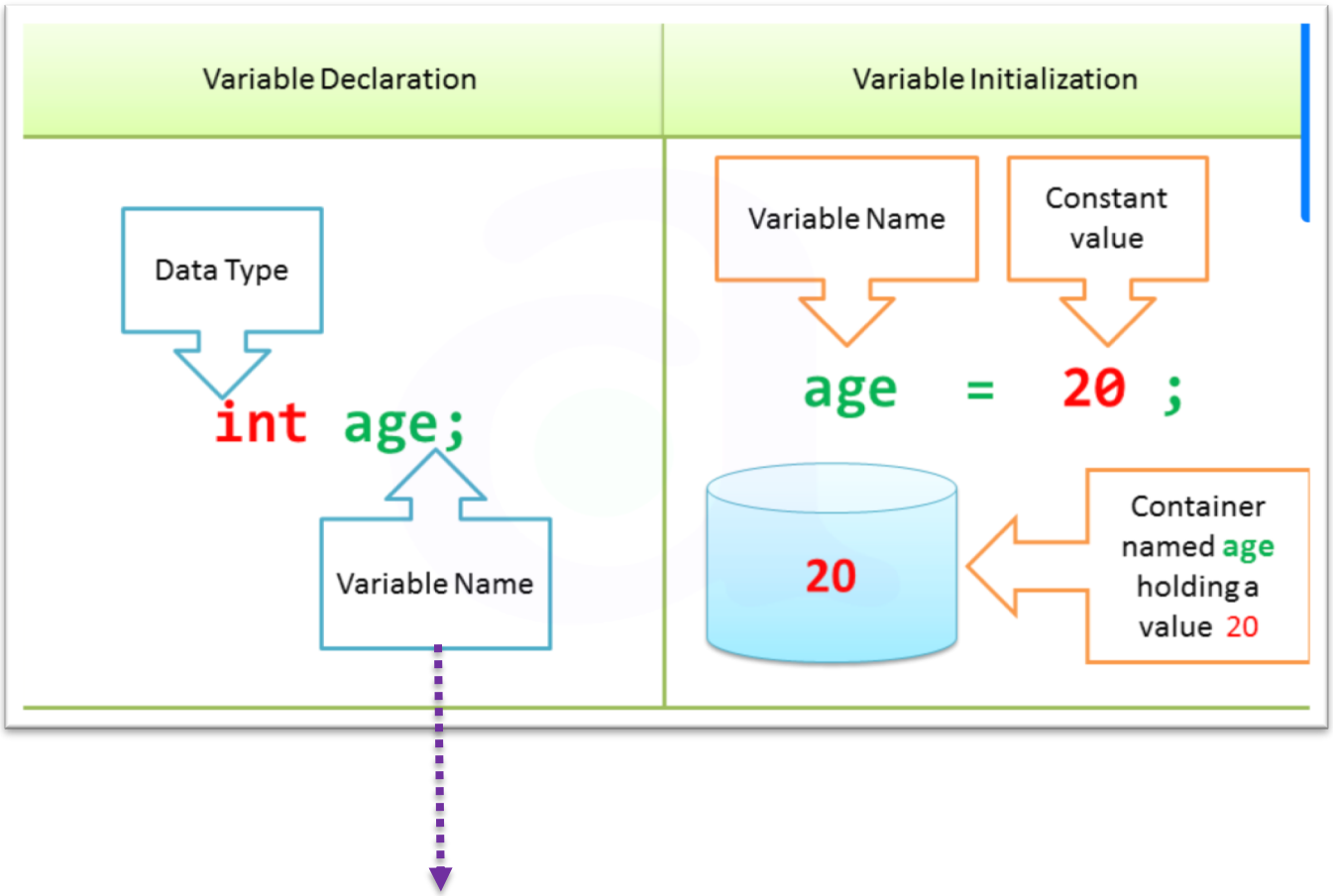
https://runestone.academy/ns/books/published/csjava/Unit1-Getting-Started/topic-1-4-assignment.html

Assignment Statements

```
public class ErrorSwap
{
  public static void main(String[] args)
  {
    int h = 3;
    int w = 5;
    System.out.println(h);  //3
    System.out.println(w);  //5
    h = w;
    w = h;
    System.out.println(h);  //expected 5
    System.out.println(w);  //expected 3
  }
}
```

https://runestone.academy/ns/books/published/csjava/Unit1-Getting-Started/topic-1-4-assignment.html

Initializing Variables



Uninitialized
(default value)

Initializing Variables

- To protect against an **uninitialized variable** (and to keep the compiler **happy**  assign a value at the time the variable is declared.
- Examples:

```
int count = 0;
char grade = 'A';
int a =10, b=20, c=30;
```

Sample Program

```
import java.util.Scanner;
public class EggBasket2
{
    public static void main (String [] args)
    {
        int numberOfBaskets, eggsPerBasket, totalEggs;
        Scanner keyboard = new Scanner (System.in);
        System.out.println ("Enter the number of eggs in each basket:");
        eggsPerBasket = keyboard.nextInt ();
        System.out.println ("Enter the number of baskets:");
        numberOfBaskets = keyboard.nextInt ();
        totalEggs = numberOfBaskets * eggsPerBasket;
        System.out.println ("If you have");
        System.out.println (eggsPerBasket + " eggs per basket and");
        System.out.println (numberOfBaskets + " baskets, then");
        System.out.println ("the total number of eggs is " + totalEggs);
        System.out.println ("Now we take two eggs out of each basket.");
        eggsPerBasket = eggsPerBasket - 2;
        totalEggs = numberOfBaskets * eggsPerBasket;
        System.out.println ("You now have");
        System.out.println (eggsPerBasket + " eggs per basket and");
        System.out.println (numberOfBaskets + " baskets.");
        System.out.println ("The new total number of eggs is " + totalEggs);
    }
}
```

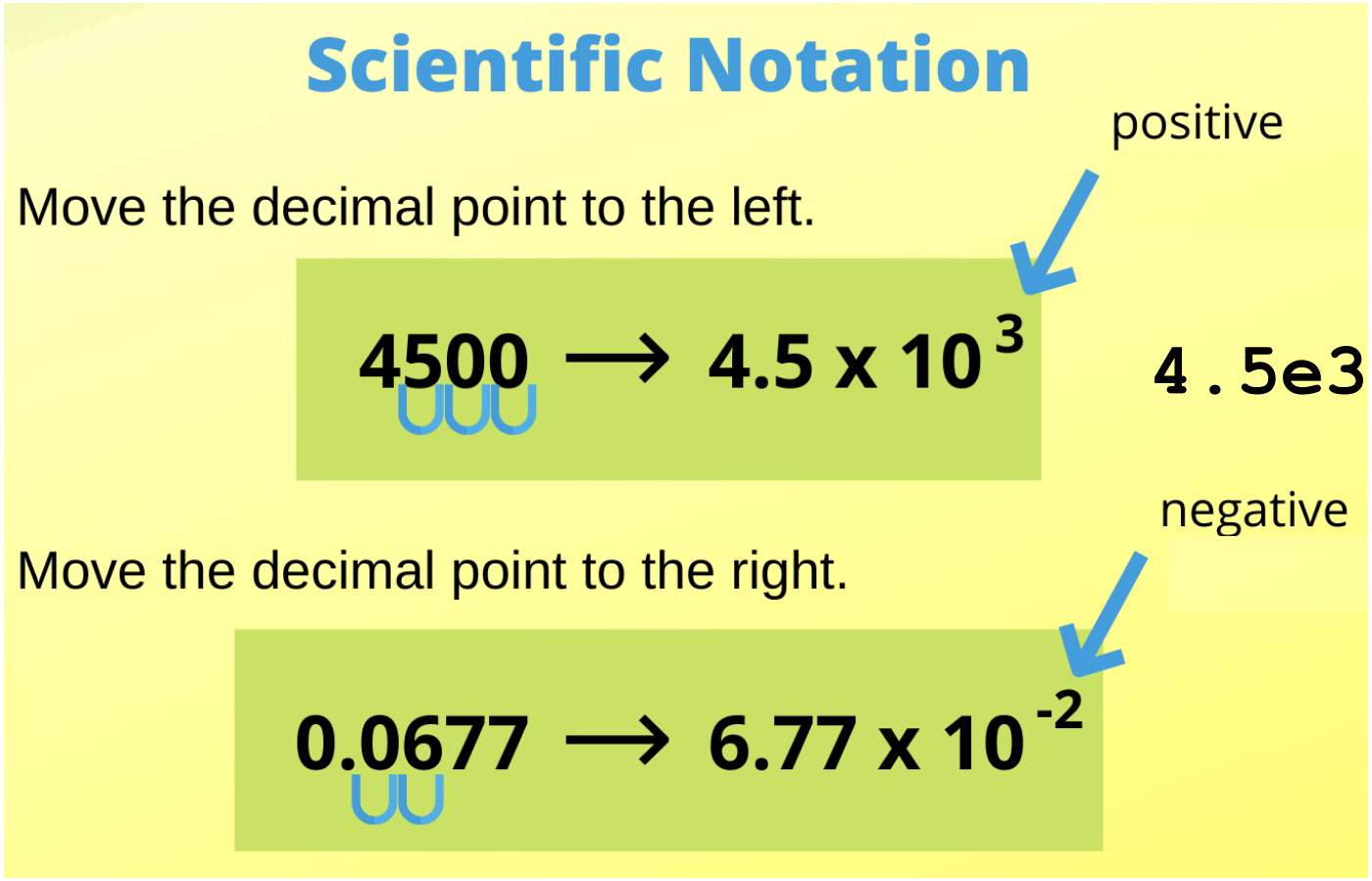
Sample Program (Output)

```
Enter the number of eggs in each basket:
6
Enter the number of baskets:
10
If you have
6 eggs per basket and
10 baskets, then
the total number of eggs is 60
Now we take two eggs out of each basket.
You now have
4 eggs per basket and
10 baskets.
The new total number of eggs is 40
```

Constants

- **Literal expressions** such as `2, 3.7,` or `'y'` are called ***constants*.**
- **Integer constants** can be preceded by a **+** or **–** sign (no commas)
  - `+2 , +199 , -5147483648 , 2.71828 , 29002, 0`
- **Floating-point** constants can be written
  - With digits after a decimal point or
  - Using *e notation.*

e Notation

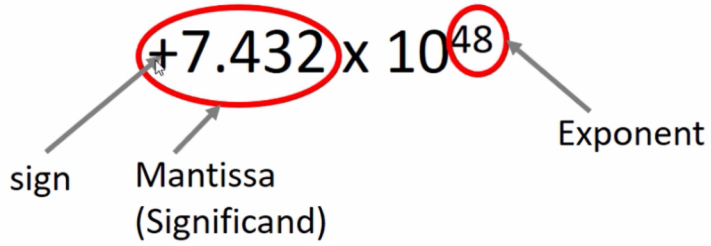


https://sciencenotes.org/scientific-notation-working-with-exponents/

Floating-point

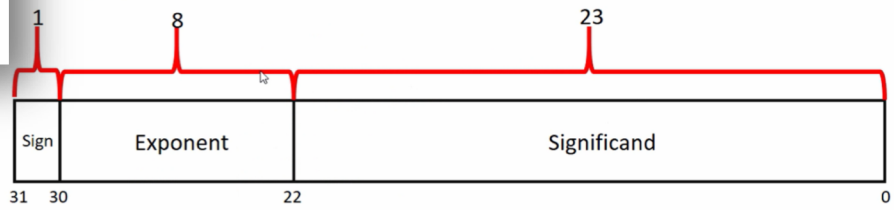
- Floating-point numbers often are only **approximations** since they are stored **with a finite number of bits**.
- `1.0/3.0` is slightly less than 1/3.
- `1.0/3.0 + 1.0/3.0 + 1.0/3.0` is less than 1.

The IEEE-754 floating-point standard



The IEEE-754 floating-point standard

- Single precision –>32 bits representation



https://fastbitlab.com/single-vs-double-precision-floating-point-representation/

Named Constants

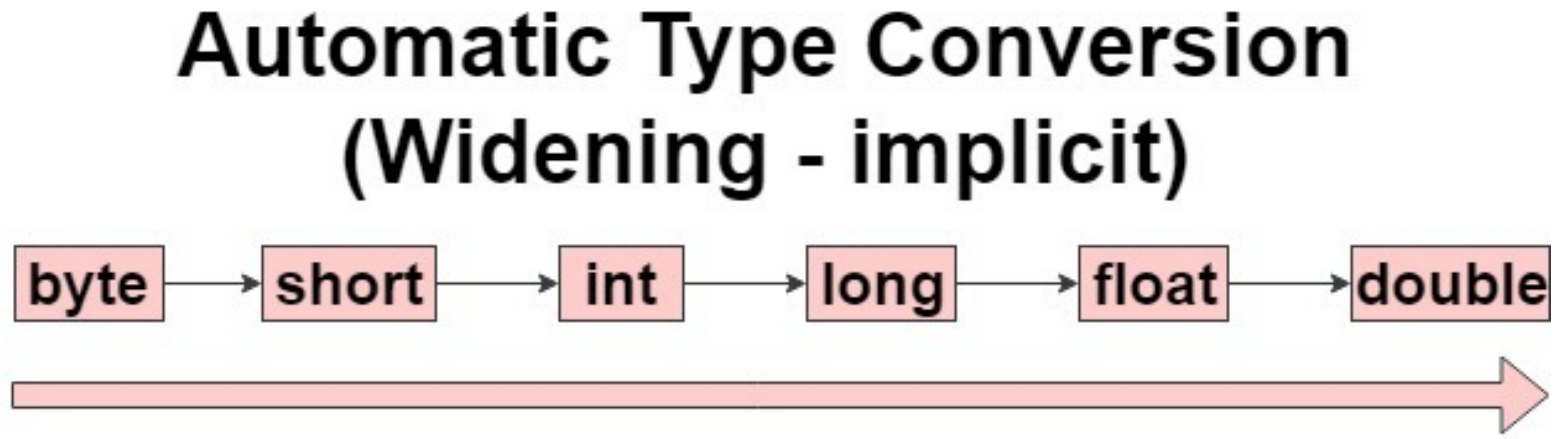
- Java provides mechanism to …
  - Define a **variable**
  - **Initialize** it
  - **Fix** the **value** so it cannot be changed

```
public static final Type Variable = Constant;
```

  - Example:

```
public static final double PI = 3.14159;
```

Type Casting (1/2)



- When we assign value of a **smaller** data type to a **bigger** data type.

Type Casting (1/2)

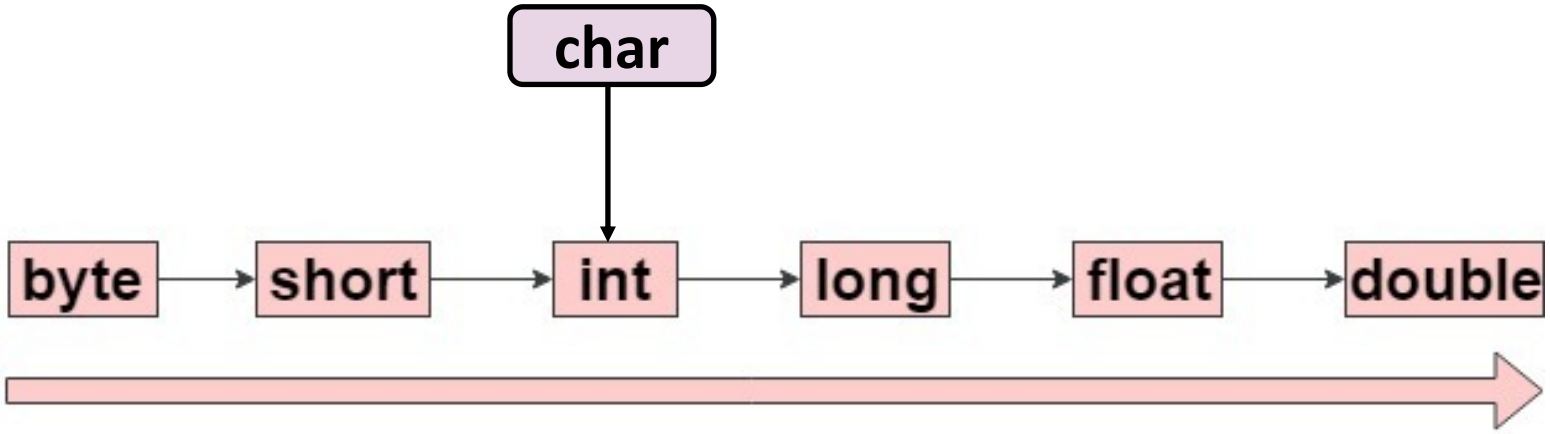
```
class Test
{
    public static void main(String[] args)
    {
        int i = 100;

        //automatic type conversion
        long l = i;

        //automatic type conversion
        float f = l;
        System.out.println("Int value "+i);
        System.out.println("Long value "+l);
        System.out.println("Float value "+f);
    }
}
```

Output

```
Int value 100
Long value 100
Float value 100.0
```

Type Casting (1/2)

- You can assign a value of type **char** to a variable of type **int**.



Type Casting (1/2)

- You can assign a value of type **char** to a variable of type **int**.

```
class CharToInt {
    public static void main(String[] args) {
        char x = '8';
        int z = x;
        System.out.println(x);
    }
}
```

Output: 56

```
class CharToInt {
    public static void main(String[] args) {
        char x = '88';
        int z = x;
        System.out.println(x);
    }
}
```

Output: Error

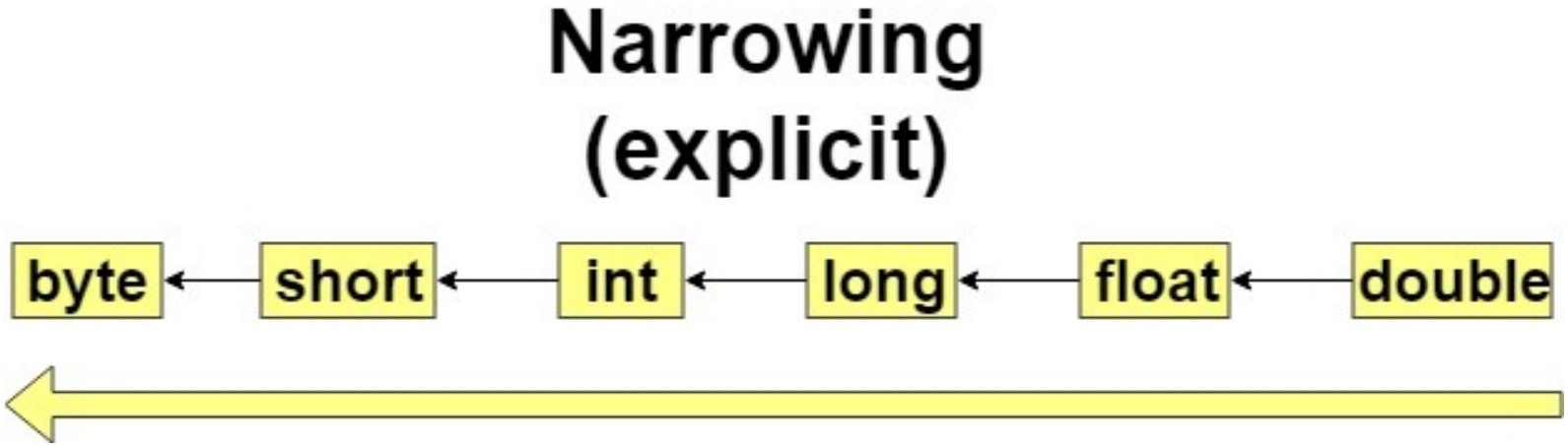
```
class CharToInt {
    public static void main(String[] args) {
        char x = 'A';
        int z = x;
        System.out.println(x);
    }
}
```

Output: 65

```
class CharToInt {
    public static void main(String[] args) {
        char x = '8';
        int z = (int)x;
        System.out.println(x);
    }
}
```

Output: 56

Type Casting (2/2)



- If we want to assign a value of **larger** data type to a **smaller** data type.

Type Casting (2/2)

```
//Java program to illustrate explicit type conversion
class Test
{
    public static void main(String[] args)
    {
        double d = 100.04;

        //explicit type casting
        long l = (long)d;

        //explicit type casting
        int i = (int)l;
        System.out.println("Double value "+d);

        //fractional part lost
        System.out.println("Long value "+l);

        //fractional part lost
        System.out.println("Int value "+i);
    }
}
```

Output

```
Double value 100.04
Long value 100
Int value 100
```

Type Casting (2/2)

```
class PrepBytes {
    public static void main(String args[])
    {
        double data = 52.6345;
        System.out.println("Double - " + data);

        int value = (int)data;

        System.out.println("Integer - " + value);
    }
}
```

[] args

Output:

```
Double - 52.6345
Integer - 52
```

- Any nonzero value to the right of the decimal point is ***truncated*** rather than ***rounded***.

Type Casting

```java
class IntToDouble {
    public static void main(String[] args) {
        int x = 11;
        int y = 3;
        int z = x/y;
        System.out.println(z);
    }
}
```

Output: 3

```java
class IntToDouble {
    public static void main(String[] args) {
        int x = 11;
        int y = 3;
        double z = x/y;
        System.out.println(z);
    }
}
```

Output: 3.0

Type Casting

```
class IntToDouble {
    public static void main(String[] args) {
        double x = 11.0;
        double y = 3.0;
        double z = x/y;
        System.out.println(z);
    }
}
```

Output: 3.6666666666666665

```
class DoubleToInt {
    public static void main(String[] args) {
        double x = 11.0;
        double y = 3.0;
        int z = (int)(x/y);
        System.out.println(z);
    }
}
```

Output: 3