Presentation 02
Input, Output and Casting in Java

Christian Rodríguez Bustos

Edited by Juan Mendivelso

Object Oriented Programming

Agenda

1. Output in Java
2. Input in Java
3. Casting

1. Output in Java

1.1 Output Methods
1.2 Escape Sequences
1.3 Integers
1.4 Floats
1.5 Chars and Strings
1.6 Dates and Times

1.1 Output Methods

`System.out` is the "standard" java output stream

This stream is already open and ready to accept output data.

Print an object

```
System.out.print(Object object);
```

Print an object using a specific format

```
System.out.printf(String format, Object object);
```

Print an object and a new line

```
System.out.println(Object object);
```

System.out is the "standard" java output stream

```
System.out.print();
```

Print without moving cursor to the next line

```
System.out.print
    print(Object obj)      void
    print(String s)        void
    print(boolean b)       void
    print(char c)          void
    print(char[] s)        void
    print(double d)        void
    print(float f)         void
    print(int i)           void
    print(long l)          void
```

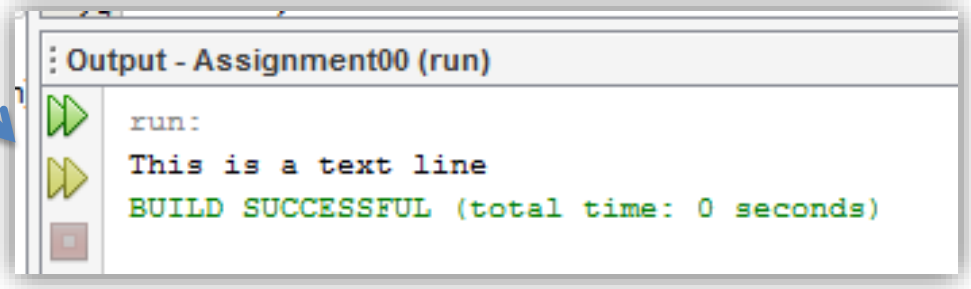
```
System.out.println
    println()              void
    println(Object x)      void
    println(String x)      void
    println(boolean x)     void
    println(char x)        void
    println(char[] x)      void
    println(double x)      void
    println(float x)       void
    println(int x)         void
    println(long x)        void
```

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

Print moving cursor to the next line

System.out is the "standard" java output stream

```
System.out.print("This is a ");
System.out.println("text line");
```



System.out is the "standard" java output stream

```
System.out.printf();
```

Print without moving cursor to the next line

```
System.out.printf
        printf(String format, Object... args) PrintStream
        printf(Locale l, String format, Ob... PrintStream
```

```
Output - Assignment00 (run)
run:
This is a text line
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
System.out.printf("%s","This is a text line");
System.out.println();
```

1.2 Escape Sequences

Escape sequences

Escape sequence	Description
`\n`	Newline. Position the screen cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor at the beginning of the current line—do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double-quote character. For example, `System.out.println( "\"in quotes\"" );` displays `"in quotes"`

Escape sequences: new line and tab examples

```
System.out.print("Line with a new line sequence escape \n");
```

Output - OOP20112 (run)

```
run:
Line with a new line sequence escape
BUILD SUCCESSFUL (total time: 1 second)
```

```
System.out.print("Line with a \t tabulation sequence escape \n");
```

Output - OOP20112 (run)

```
run:
Line with a      tabulation sequence escape
BUILD SUCCESSFUL (total time: 1 second)
```

Escape sequences: slash and double quote examples

```
System.out.print("Line with a \\ slash sequence escape \n");
```

```
Output - OOP20112 (run)
run:
Line with a \ slash sequence escape
BUILD SUCCESSFUL (total time: 1 second)
```

```
System.out.print("Line with a \" double quote sequence escape ");
```

```
Output - OOP20112 (run)
run:
Line with a " double quote sequence escape
BUILD SUCCESSFUL (total time: 1 second)
```

1.3 Integers

Printing formats: Numbers with zeros

```java
public class FormatOutputExample {

    public static void main(String[] args) {

        System.out.printf("Num is %03d\n", 5);

    }
}
```

```
Output - OOP20112 (run)
run:
Num is 005
BUILD SUCCESSFUL (total time: 3 seconds)
```

```
"Num is %03d\n"
```

This is the **format**

Printing Integers Numbers

```java
private static void printIntegerExample() {

    int number = 26;

    System.out.printf("Printing integer: %d\n", number);
    System.out.printf("Printing positive integer: %d\n", +number);
    System.out.printf("Printing negative integer: %d\n", -number);
    System.out.printf("Printing octal integer: %o\n", number);
    System.out.printf("Printing hexadecimal integer: %x\n", number);
    System.out.printf("Printing hexadecimal integer: %X\n", number);

    System.out.printf("\nPrinting integer justified: %4d\n", 1);
    System.out.printf("Printing integer justified: %4d\n", 12);
    System.out.printf("Printing integer justified: %4d\n", 123);
    System.out.printf("Printing integer justified: %4d\n", 1234);
    System.out.printf("Printing integer justified: %4d\n", 12345);
    System.out.printf("Printing integer justified filled with zeros: %09d\n", 12345);
}
```

Printing Integers Numbers

Output - Assignment00 (run)

```
run:

Integers Formats
Printing integer: 26
Printing positive integer: 26
Printing negative integer: -26
Printing octal integer: 32
Printing hexadecimal integer: 1a
Printing hexadecimal integer: 1A

Printing integer justified:    1
Printing integer justified:   12
Printing integer justified:  123
Printing integer justified: 1234
Printing integer justified: 12345
Printing integer justified filled with zeros: 000012345
```

1.4 Floats

Printing formats: Floats

```java
public class FormatOutputExample {

    public static void main(String[] args) {

        System.out.printf("The first two PI decimals are: %.2f\n", Math.PI);
    }
}
```

Output - OOP20112 (run)

```
run:
The first two PI digits are: 3.14
BUILD SUCCESSFUL (total time: 1 second)
```

```java
"The first two PI decimals are: %.2f\n"
```

This is the
format

Printing Floating-Point Numbers

```java
private static void printFloatingPointExample() {

    double number = 12345678.9;

    System.out.printf("Printing float: %f\n", number);
    System.out.printf("Printing float in exponential notation: %e\n", +number);
    System.out.printf("Printing float in exponential notation: %E\n", +number);
    System.out.printf("Printing hexadecimal float: %A\n", number);

    System.out.printf("\nPrinting float using precision: %.3f\n", number);
    System.out.printf("Printing float using precision: %.3e\n", number);
}
```

Printing Floating-Point Numbers

Output - Assignment00 (run)

```
Floating Point Formats
Printing float: 12345678,900000
Printing float in exponential notation: 1.234568e+07
Printing float in exponential notation: 1.234568E+07
Printing hexadecimal float: 0X1.78C29DCCCCCCDP23

Printing float using precision: 12345678,900
Printing float using precision: 1.235e+07
```

1.5 Chars and Strings

Printing formats: Letters

```java
public class FormatOutputExample {

    public static void main(String[] args) {

        System.out.printf("Char values is %c\n", 'c');

    }
}
```

```java
"Char values is %c\n"
```

This is the **format**

```
Output - OOP20112 (run)
run:
Char values is c
BUILD SUCCESSFUL (total time: 0 seconds)
```

Printing Strings and characters

```java
private static void printStringsExample() {

    char letter = 'a';
    String string = "My String";
    int number = 12345;

    System.out.printf("Printing char: %c\n", letter);
    System.out.printf("Printing String: %s\n", string);
    System.out.printf("Printing Uppercase String: %S\n", string);
    System.out.printf("Printing Integer as String: %s\n", number);

    System.out.printf("\nPrinting String using precision: %.6s\n", string);
    System.out.printf("\nPrinting String using left justification: %10s%10d%10c\n",
            string,number,letter);
}
```

Printing Strings and characters

Output - Assignment00 (run)

```
String Formats
Printing char: a
Printing String: My String
Printing Uppercase String: MY STRING
Printing Integer as String: 12345

Printing String using precision: My Str

Printing String using left justification:  My String     12345         a
```

1.6 Dates and Times

Printing Dates and times

```java
private static void printDateAndTimeExample() {

    Calendar dateTime = Calendar.getInstance();

    System.out.printf("Printing date in long format: %tc\n", dateTime);
    System.out.printf("Printing date in yyyy-mm-dd format: %tF\n", dateTime);
    System.out.printf("Printing date in dd/mm/yy format: %tD\n", dateTime);
    System.out.printf("Printing time in 12 hours format: %tr\n", dateTime);
    System.out.printf("Printing time in 24 hours format: %tT\n", dateTime);
}
```

Printing Dates and times

Output - Assignment00 (run)

```
Date Time Formats
Printing date in long format: mié feb 16 23:59:09 COT 2011
Printing date in yyyy-mm-dd format: 2011-02-16
Printing date in dd/mm/yy format: 02/16/11
Printing time in 12 hours format: 11:59:09 PM
Printing time in 24 hours format: 23:59:09
BUILD SUCCESSFUL (total time: 0 seconds)
```

Printing formats: Dates

Remember to import the **Date Class**

```
import java.util.Date;

public class FormatOutputExample {

    public static void main(String[] args) {

        Date date = new Date();
        System.out.printf("The date is %s\n", date);
    }
}
```

```
"The date is %s\n"
```

This is the **format**

```
Output - OOP20112 (run)
run:
The date is Mon Aug 08 18:24:01 COT 2011
BUILD SUCCESSFUL (total time: 2 seconds)
```

Printing formats resources

See more formats on

http://www.java2s.com/Code/JavaAPI/java.lang/System.out.printf.htm

2. Input in Java

System.in is used to read user inputs

System.in and **Scanner** class allow us to read values typed by the user

```
import java.util.Scanner;

public class UserInputReaderExample {
    // ...
```

First we need to import the **Scanner class** at the beginning of our source code file

System.in: reading strings example

Creating the scanner

Reading an integer

```
3   import java.util.Scanner;
4
5   public class UserInputReaderExample {
6
7       public static void main(String[] args) {
8
9           Scanner reader = new Scanner(System.in);
10
11          int age;
12
13          System.out.print("Please enter your age: ");
14          age = reader.nextInt();
15
16          System.out.println("Your age is " + age);
17      }
18  }
```

Output - OOP20112 (run)

```
run:
Please enter your age: 1000
Your age is 1000
BUILD SUCCESSFUL (total time: 6 seconds)
```

System.in: reading strings example

Creating the scanner

Reading a String

```
3   import java.util.Scanner;
4
5   public class UserInputReaderExample {
6
7       public static void main(String[] args) {
8
9           Scanner reader = new Scanner(System.in);
10
11          String name;
12
13          System.out.print("Please enter your name: ");
14          name = reader.nextLine();
15
16          System.out.println("Your name is " + name);
17      }
18  }
```

Output - OOP20112 (run)

```
run:
Please enter your name: OOP Man
Your name is OOP Man
BUILD SUCCESSFUL (total time: 13 seconds)
```

System.in: reading posibilities

```
nextBigDecimal()            BigDecimal
nextBigInteger()            BigInteger
nextBigInteger(int radix)   BigInteger
nextBoolean()               boolean
nextByte()                  byte
nextByte(int radix)         byte
nextDouble()                double
nextFloat()                 float
nextInt()                   int
nextInt(int radix)          int
nextLong()                  long
nextLong(int radix)         long
nextShort()                 short
nextShort(int radix)        short
```

3. Casting

Primitive Types Casting

Java Primitive Types

Type	Size in bits	Values	Standard
`boolean`		`true` or `false`	
[*Note:* A `boolean`'s representation is specific to the Java Virtual Machine on each platform.]			
`char`	16	`'\u0000'` to `'\uFFFF'` (0 to 65535)	(ISO Unicode character set)
`byte`	8	–128 to +127 (-2^7 to $2^7 - 1$)	
`short`	16	–32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	
`int`	32	–2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	
`long`	64	–9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (-2^{63} to $2^{63} - 1$)	
`float`	32	*Negative range:* –3.4028234663852886E+38 to –1.40129846432481707e–45 *Positive range:* 1.40129846432481707e–45 to 3.4028234663852886E+38	(IEEE 754 floating point)
`double`	64	*Negative range:* –1.7976931348623157E+308 to –4.94065645841246544e–324 *Positive range:* 4.94065645841246544e–324 to 1.7976931348623157E+308	(IEEE 754 floating point)

Primitive Types Casting

		Assignation variable							
		int	long	float	double	char	byte	short	boolean
Value to assign	int	-	A	A	A	C	C	C	N
	long	C	-	A	A	C	C	C	N
	float	C	C	-	A	C	C	C	N
	double	C	C	C	-	C	C	C	N
	char	A	A	A	A	-	C	C	N
	byte	A	A	A	A	C	-	A	N
	short	A	A	A	A	C	C	-	N
	boolean	N	N	N	N	N	N	N	-

C = Explicit Cast Required

A = Automatic Cast

Casting example

```java
public static void castingExample() {

    char character = 'A';
    int integer = 100;

    System.out.println("Character value: " + character + " integer value: " + integer);

    // Automatic Cast
    integer = character;

    System.out.println("Character value: " + character + " integer value: " + integer);

    integer = 100;

    // Explicit Cast
    character = (char) integer;

    System.out.println("Character value: " + character + " integer value: " + integer);

}
```

Output - Assignment00 (run)

```
run:
Character value: A integer value: 100
Character value: A integer value: 65
Character value: d integer value: 100
BUILD SUCCESSFUL (total time: 0 seconds)
```

ASCII table

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

References

- [Barker] J. Barker, *Beginning Java Objects: From Concepts To Code*, Second Edition, Apress, 2005.
- [Deitel] H.M. Deitel and P.J. Deitel, *Java How to Program: Early Objects Version*, Prentice Hall, 2009.
- [Sierra] K. Sierra and B. Bates, *Head First Java*, 2nd Edition, O'Reilly Media, 2005.
- **Code Conventions for the Java Programming Language**, available at http://java.sun.com/docs/codeconv/CodeConventions.pdf