Java Basics II

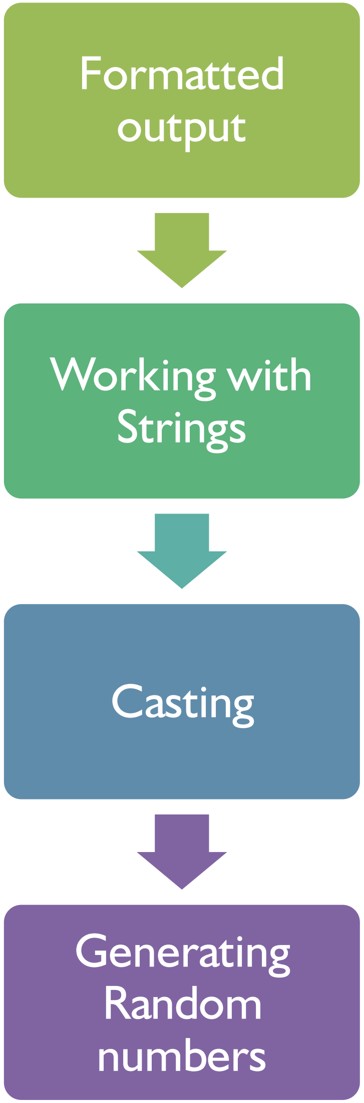
Christian Rodríguez Bustos

Object Oriented Programming

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ

un swe

Agenda



Formatted output

Working with Strings

Casting

Generating Random numbers

Formatted Output

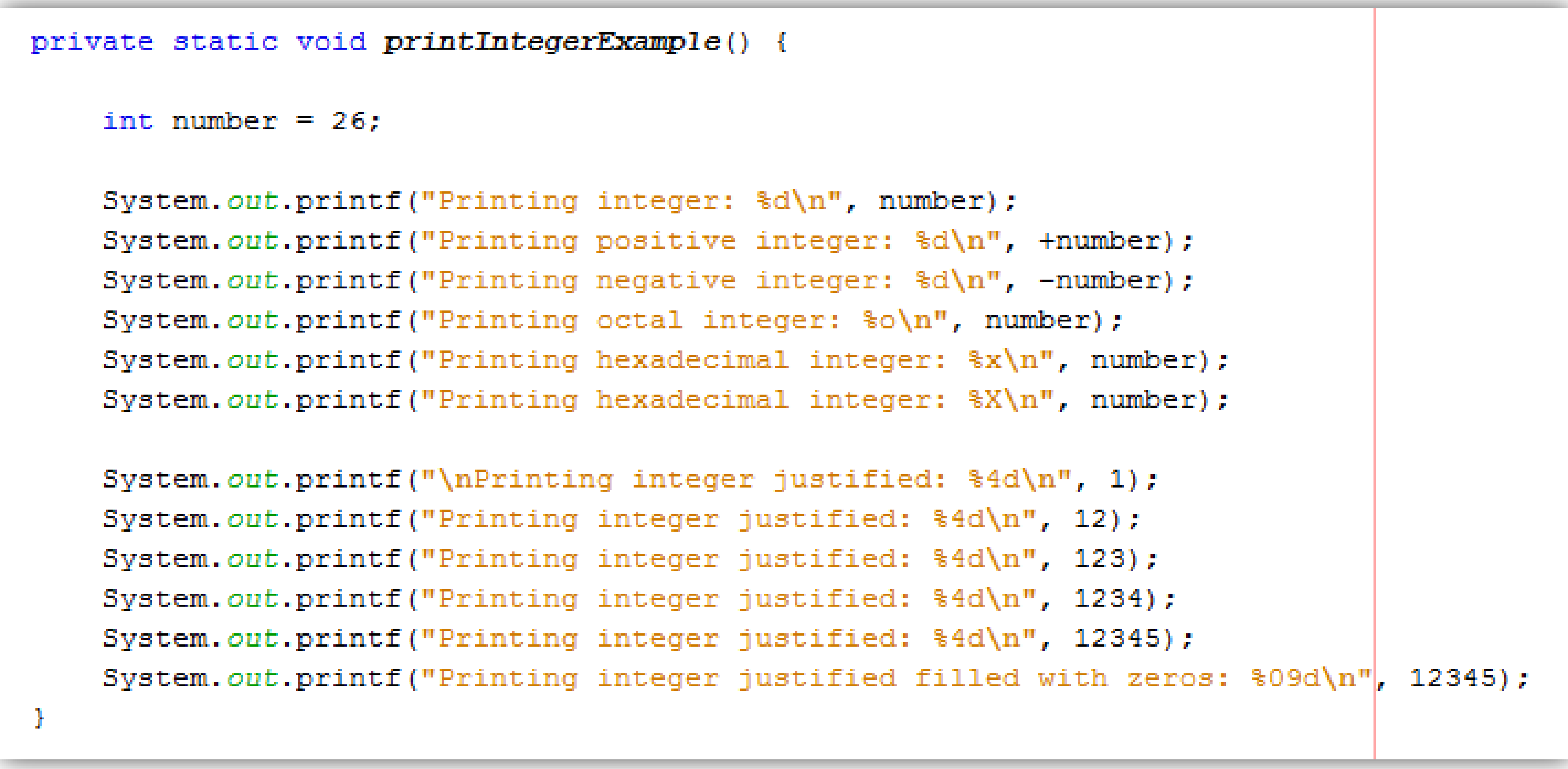
[Deitel] Chapter 28

Printing Integers & Floating-Point Numbers

Printing Strings and characters

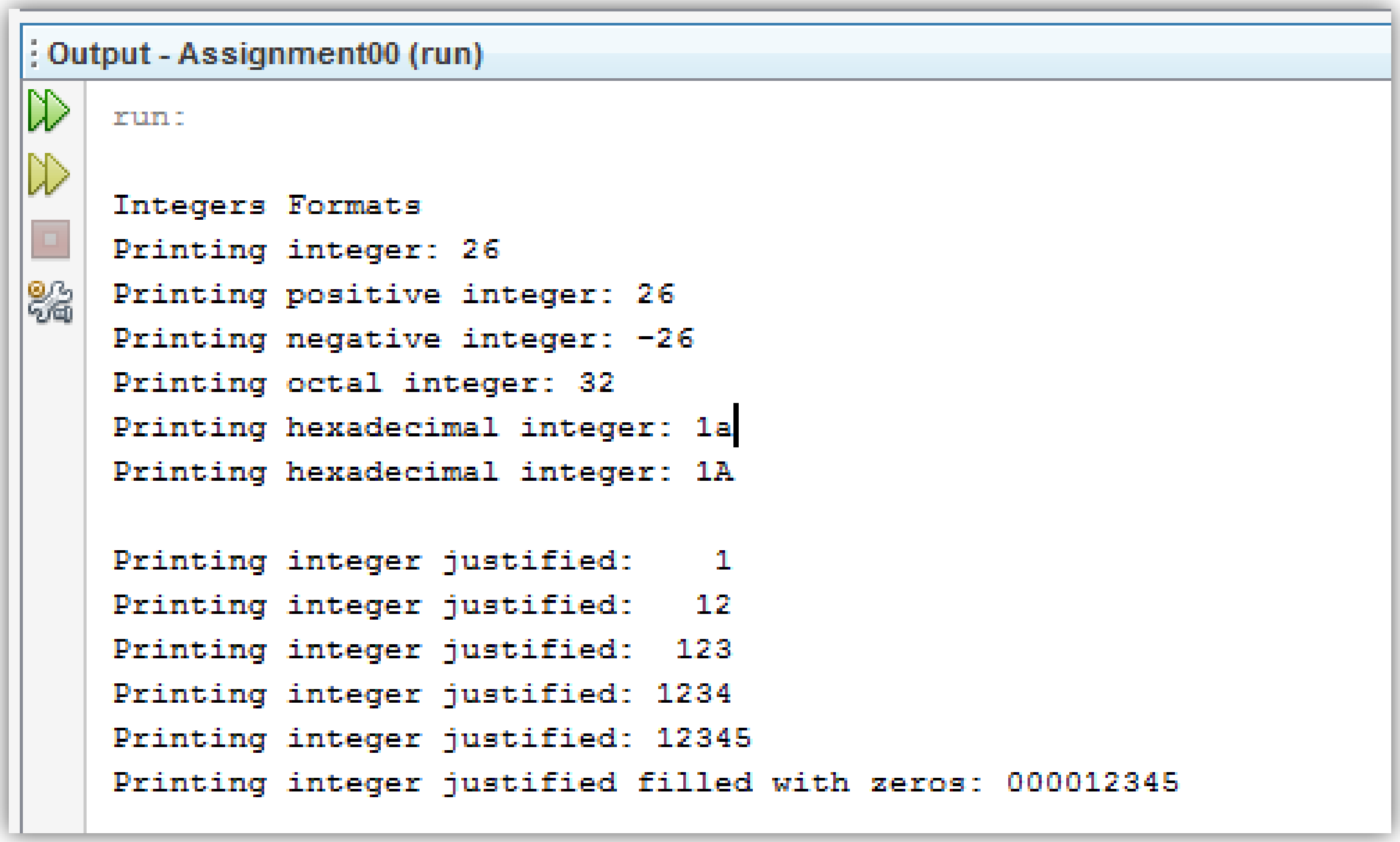
Printing Dates and times

Printing Integers Numbers



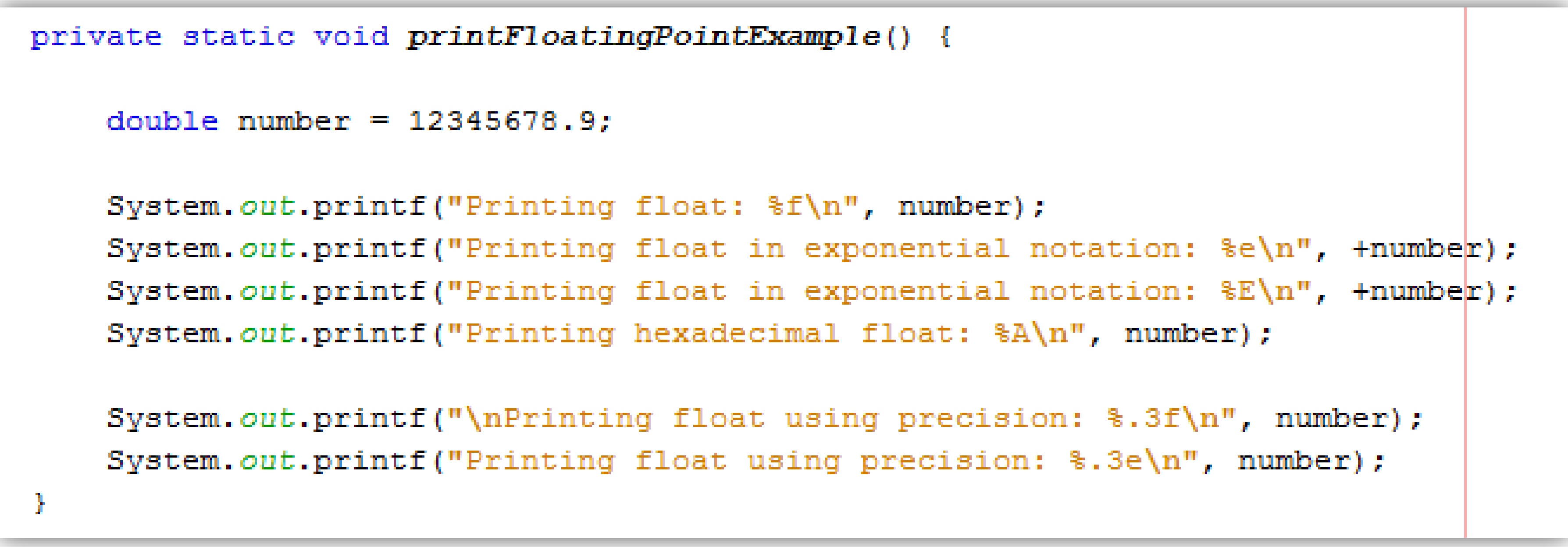
```java
private static void printIntegerExample() {

    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



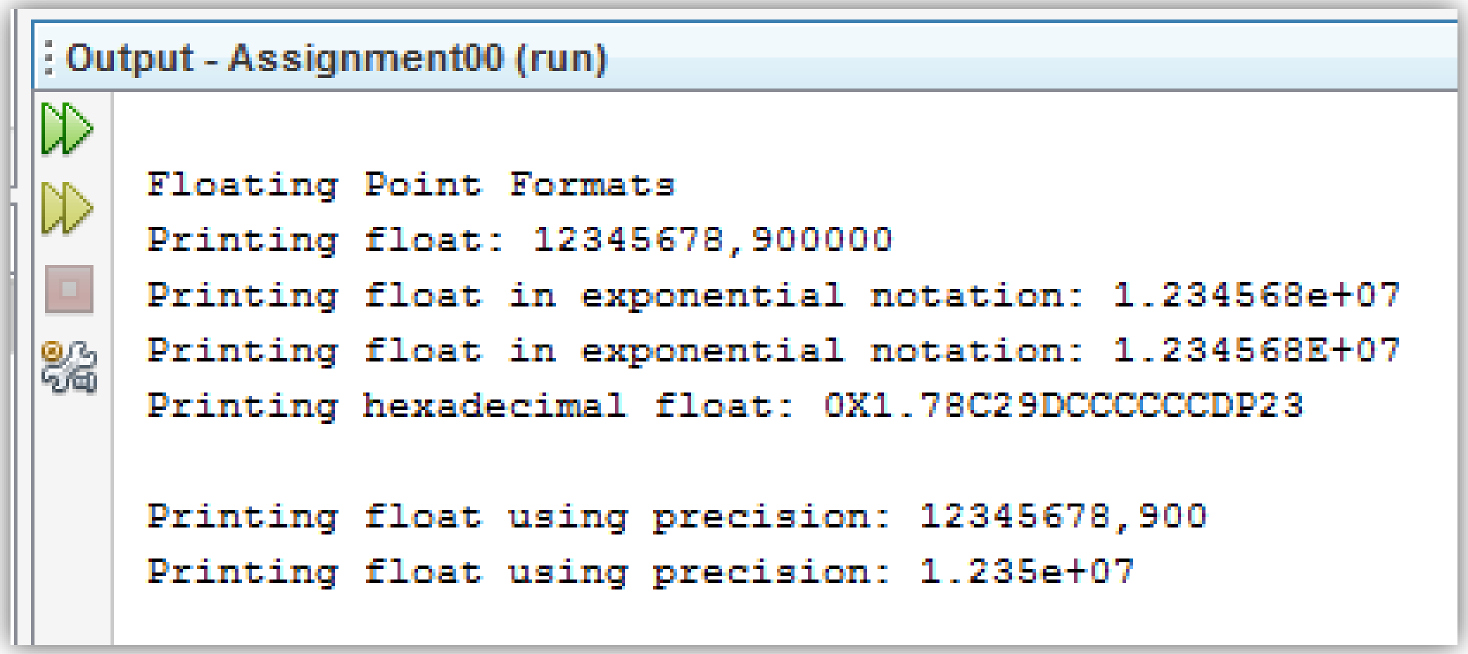
```
run:

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
```

Printing Floating-Point Numbers



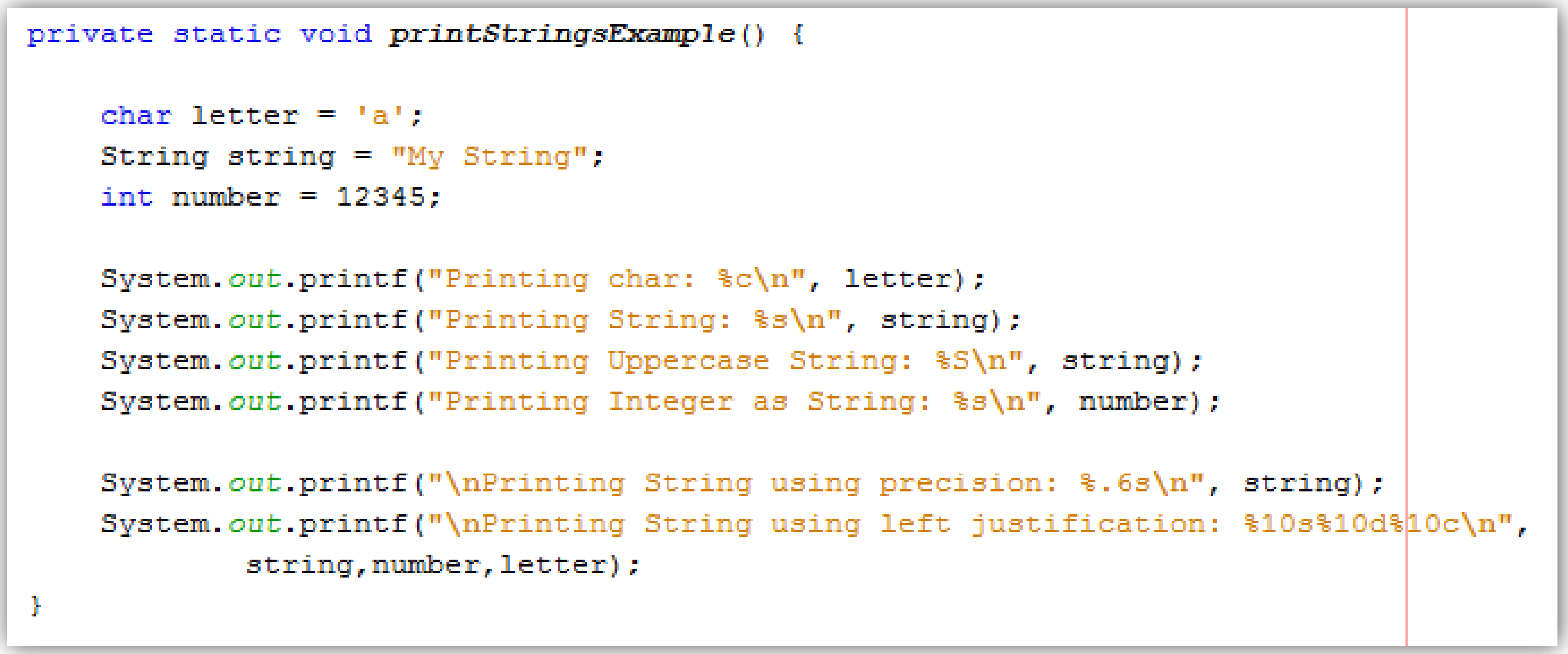
```java
private static void printFloatingPointExample() {

    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



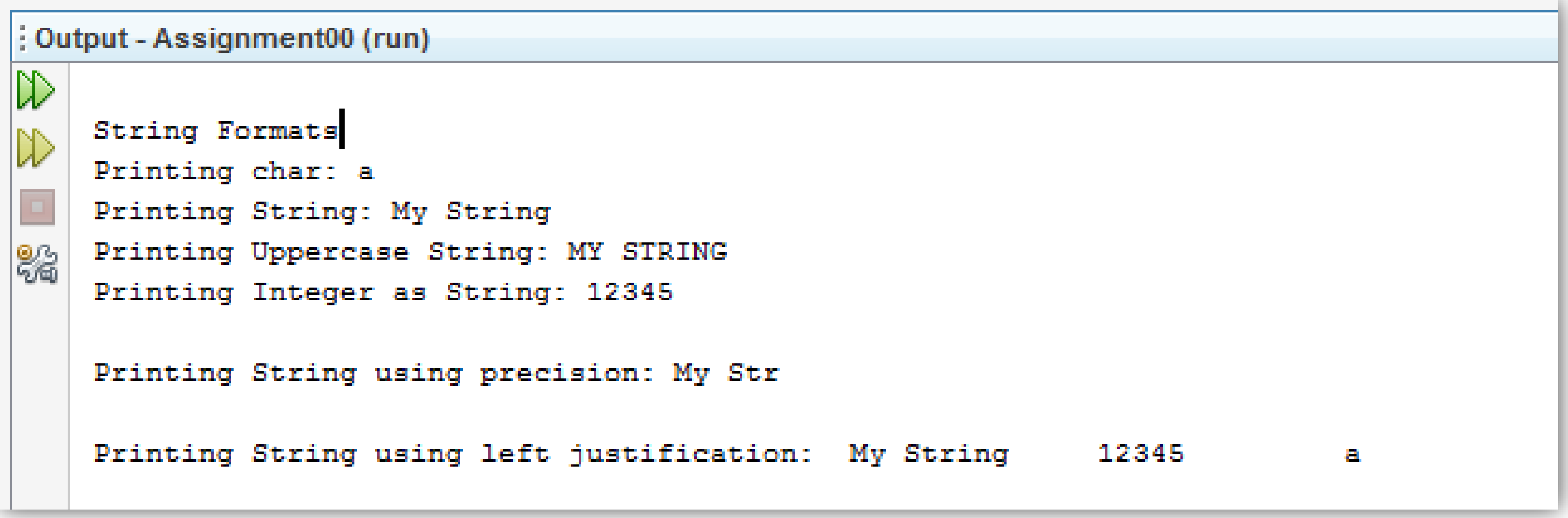
```
Floating Point Formats
Printing float: 12345678,900000
Printing float in exponential notation: 1.234568e+07
Printing float in exponential notation: 1.234568E+07
Printing hexadecimal float: 0X1.78C29DCCCCCCDP23

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

Printing Strings and characters



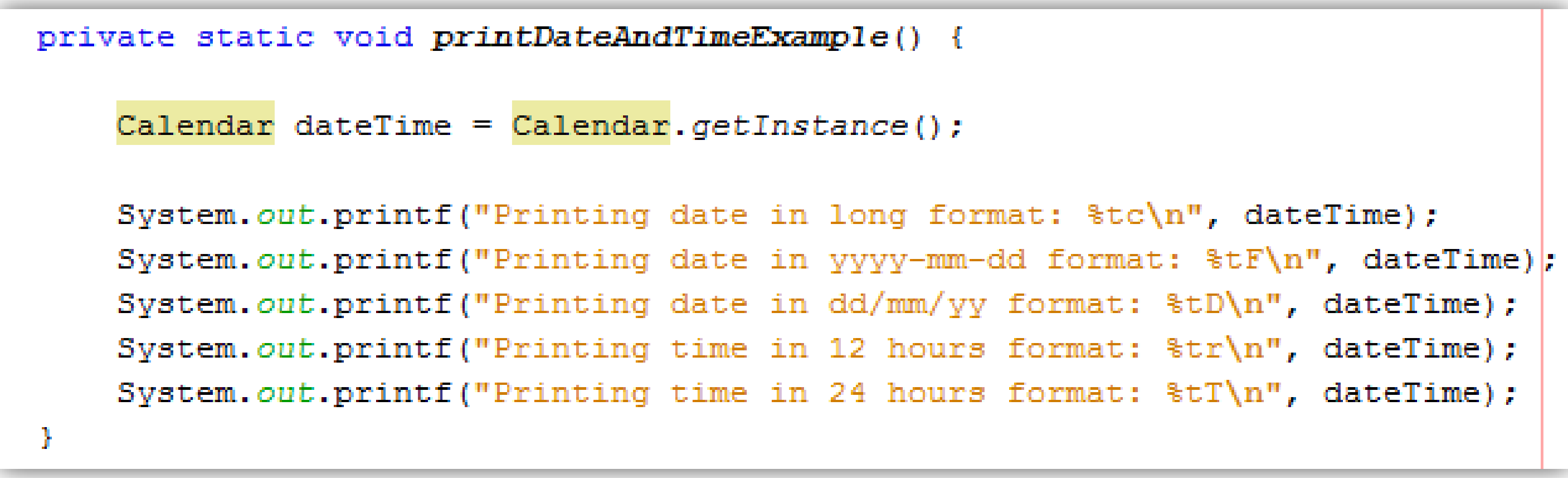
```java
private static void printStringsExample() {

    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



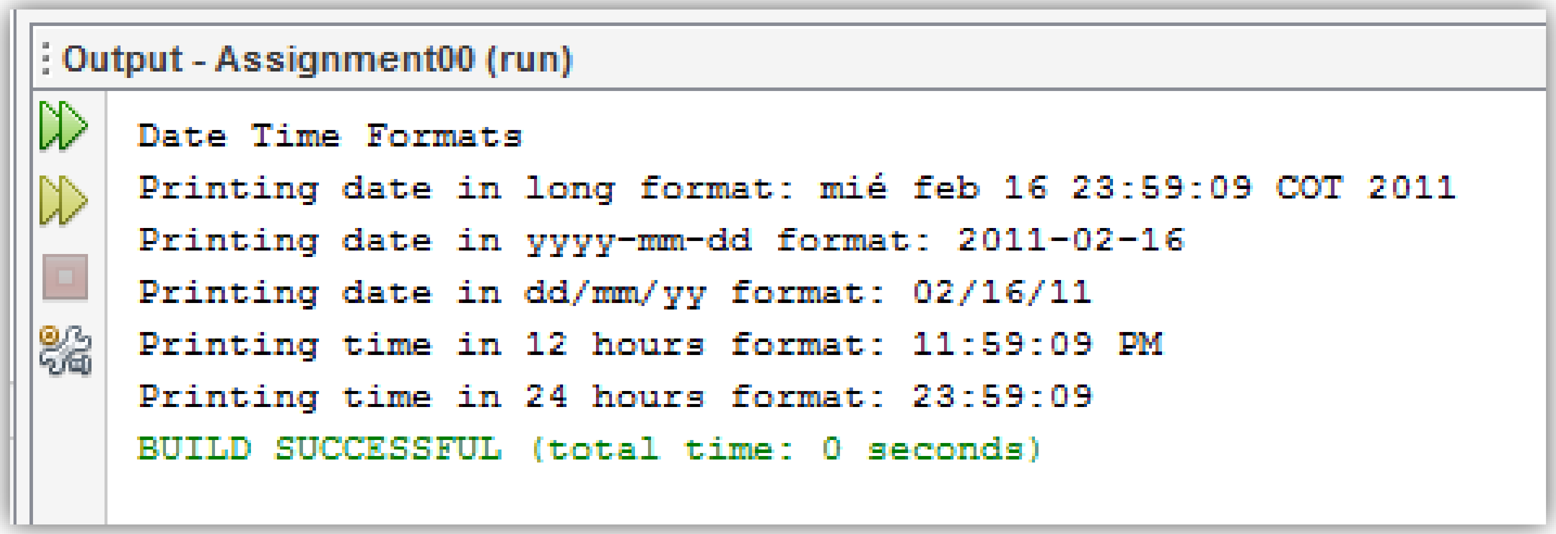
```

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
```

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



```
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)
```

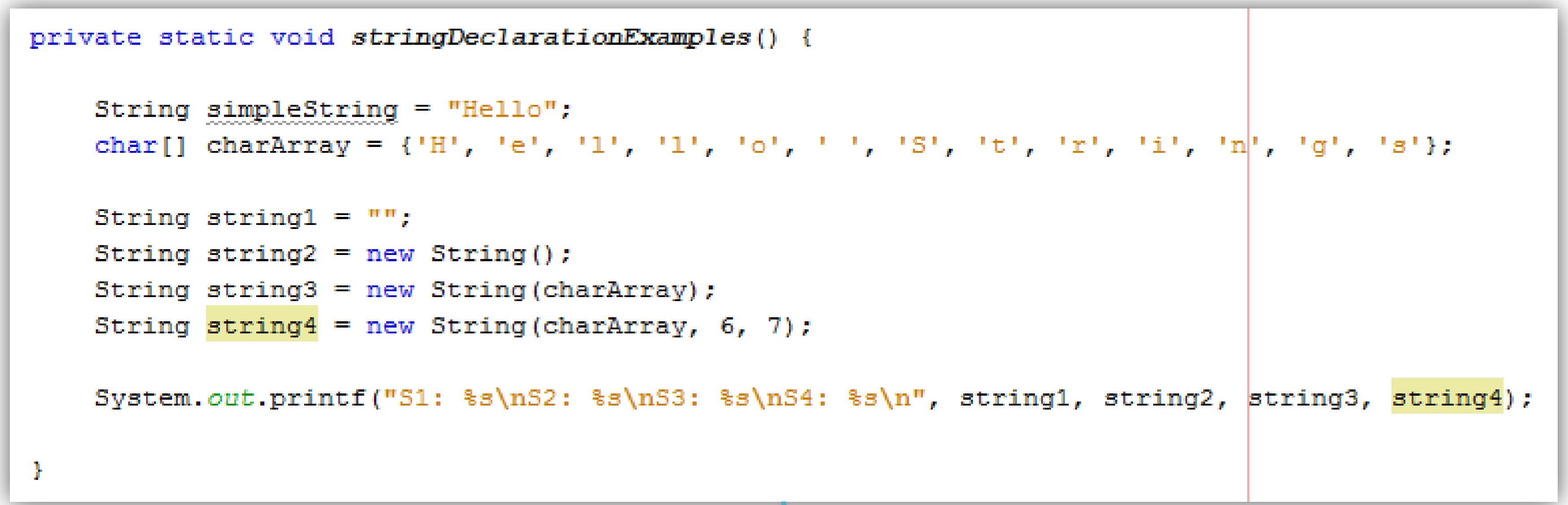
Working with Strings

[Deitel] Chapter 29

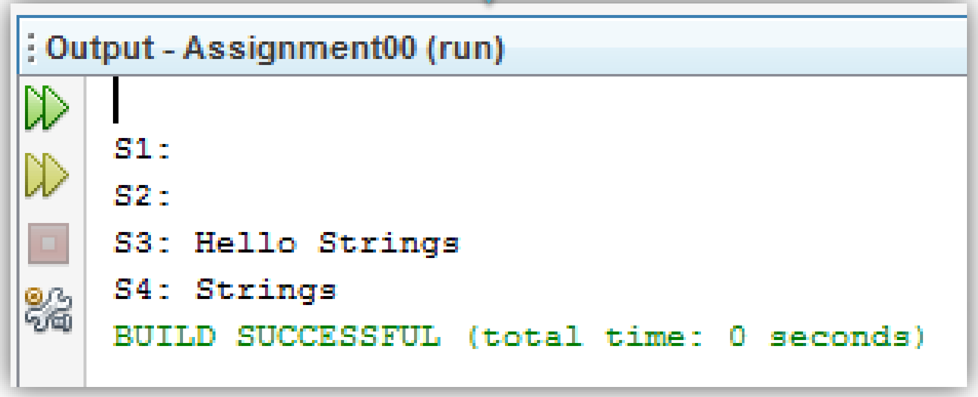
Strings Declarations

String Miscellaneous

Strings Declarations

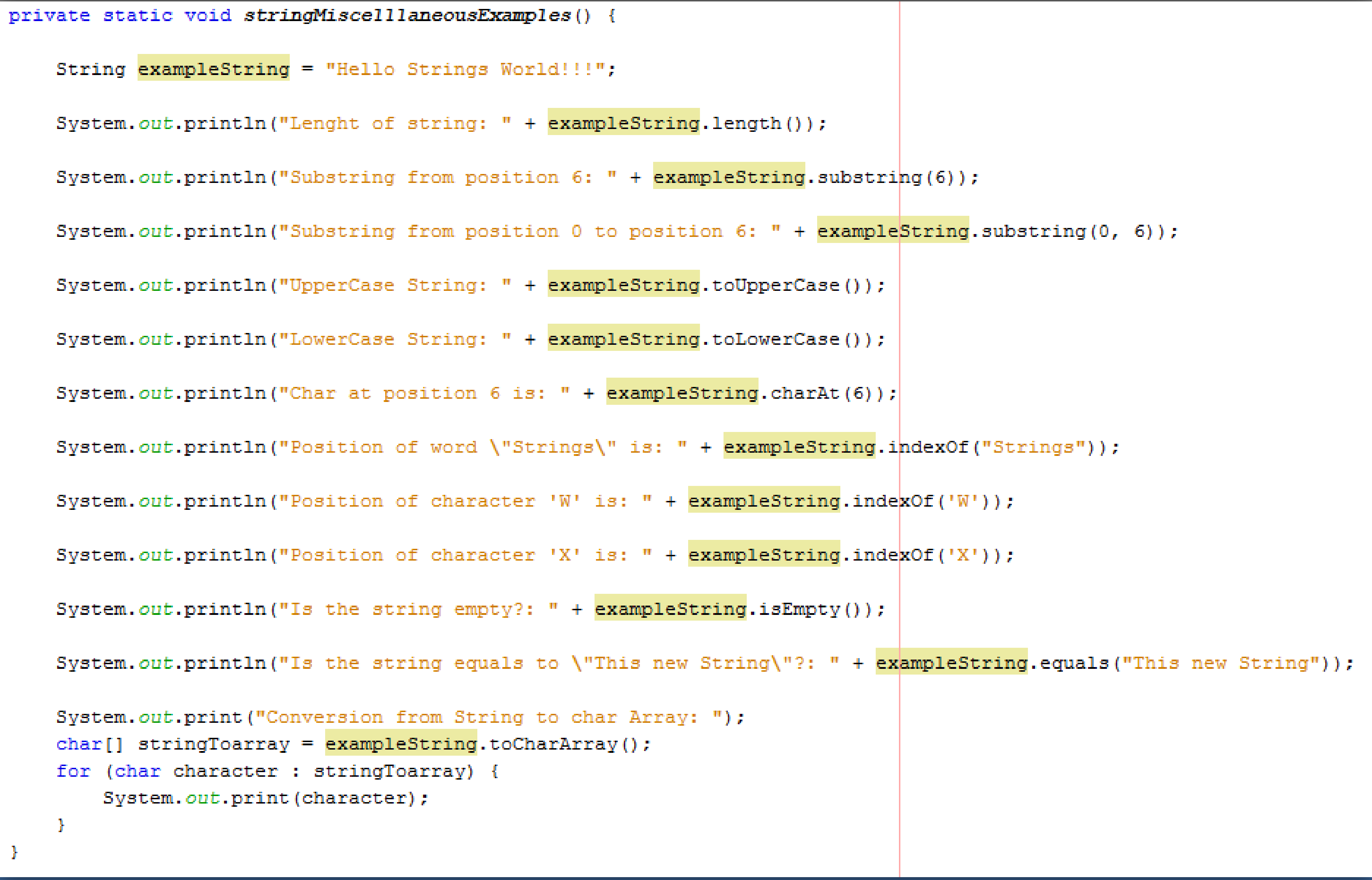


```java
private static void stringDeclarationExamples() {

    String simpleString = "Hello";
    char[] charArray = {'H', 'e', 'l', 'l', 'o', ' ', 'S', 't', 'r', 'i', 'n', 'g', 's'};

    String string1 = "";
    String string2 = new String();
    String string3 = new String(charArray);
    String string4 = new String(charArray, 6, 7);

    System.out.printf("S1: %s\nS2: %s\nS3: %s\nS4: %s\n", string1, string2, string3, string4);

}
```



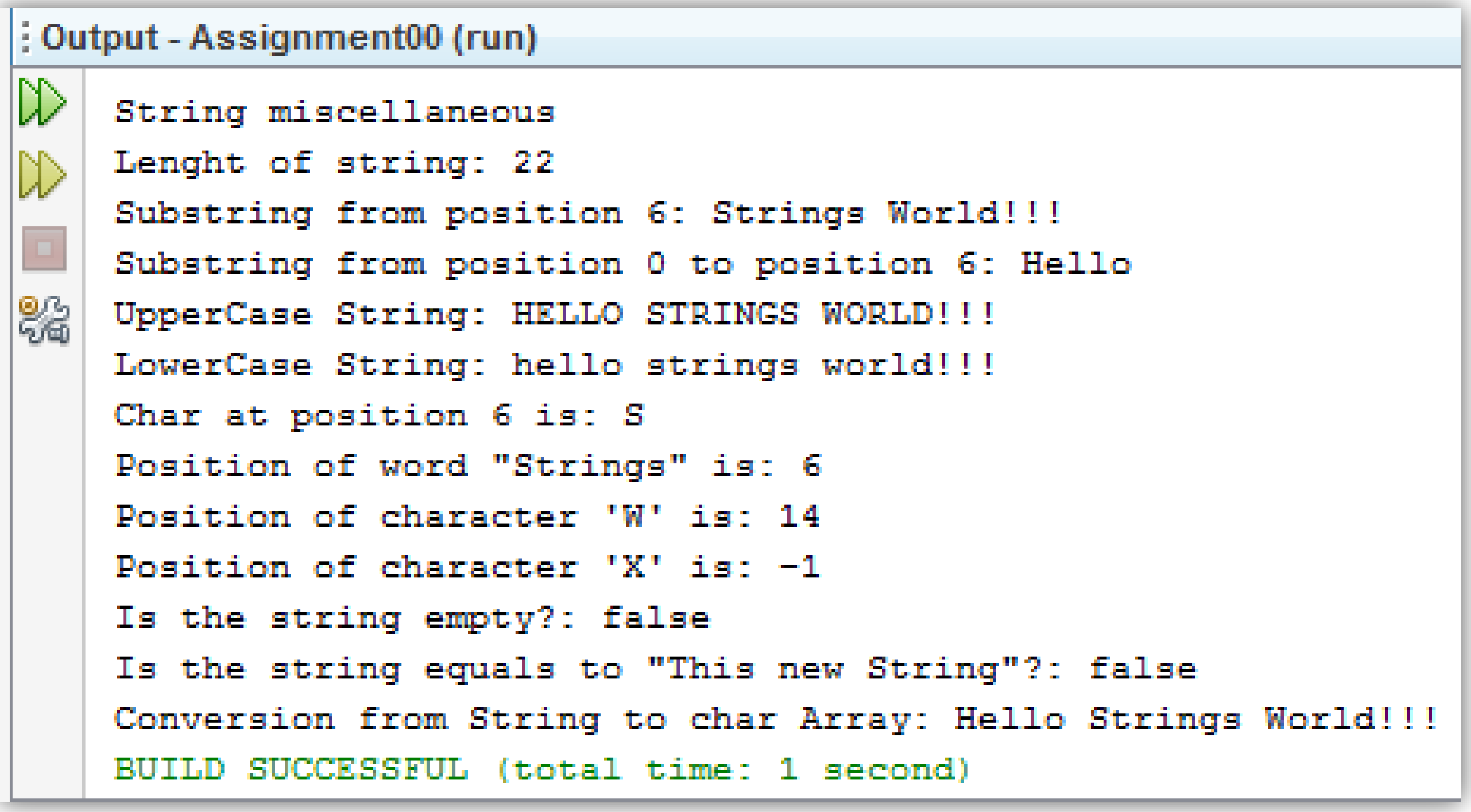
```
S1:
S2:
S3: Hello Strings
S4: Strings
BUILD SUCCESSFUL (total time: 0 seconds)
```

String Miscellaneous



```java
private static void stringMiscelllaneousExamples() {

    String exampleString = "Hello Strings World!!!";

    System.out.println("Lenght of string: " + exampleString.length());

    System.out.println("Substring from position 6: " + exampleString.substring(6));

    System.out.println("Substring from position 0 to position 6: " + exampleString.substring(0, 6));

    System.out.println("UpperCase String: " + exampleString.toUpperCase());

    System.out.println("LowerCase String: " + exampleString.toLowerCase());

    System.out.println("Char at position 6 is: " + exampleString.charAt(6));

    System.out.println("Position of word \"Strings\" is: " + exampleString.indexOf("Strings"));

    System.out.println("Position of character 'W' is: " + exampleString.indexOf('W'));

    System.out.println("Position of character 'X' is: " + exampleString.indexOf('X'));

    System.out.println("Is the string empty?: " + exampleString.isEmpty());

    System.out.println("Is the string equals to \"This new String\"?: " + exampleString.equals("This new String"));

    System.out.print("Conversion from String to char Array: ");
    char[] stringToarray = exampleString.toCharArray();
    for (char character : stringToarray) {
        System.out.print(character);
    }
}
```

String Miscellaneous



```
String miscellaneous
Lenght of string: 22
Substring from position 6: Strings World!!!
Substring from position 0 to position 6: Hello
UpperCase String: HELLO STRINGS WORLD!!!
LowerCase String: hello strings world!!!
Char at position 6 is: S
Position of word "Strings" is: 6
Position of character 'W' is: 14
Position of character 'X' is: -1
Is the string empty?: false
Is the string equals to "This new String"?: false
Conversion from String to char Array: Hello Strings World!!!
BUILD SUCCESSFUL (total time: 1 second)
```

Casting

Primitive Types Casting

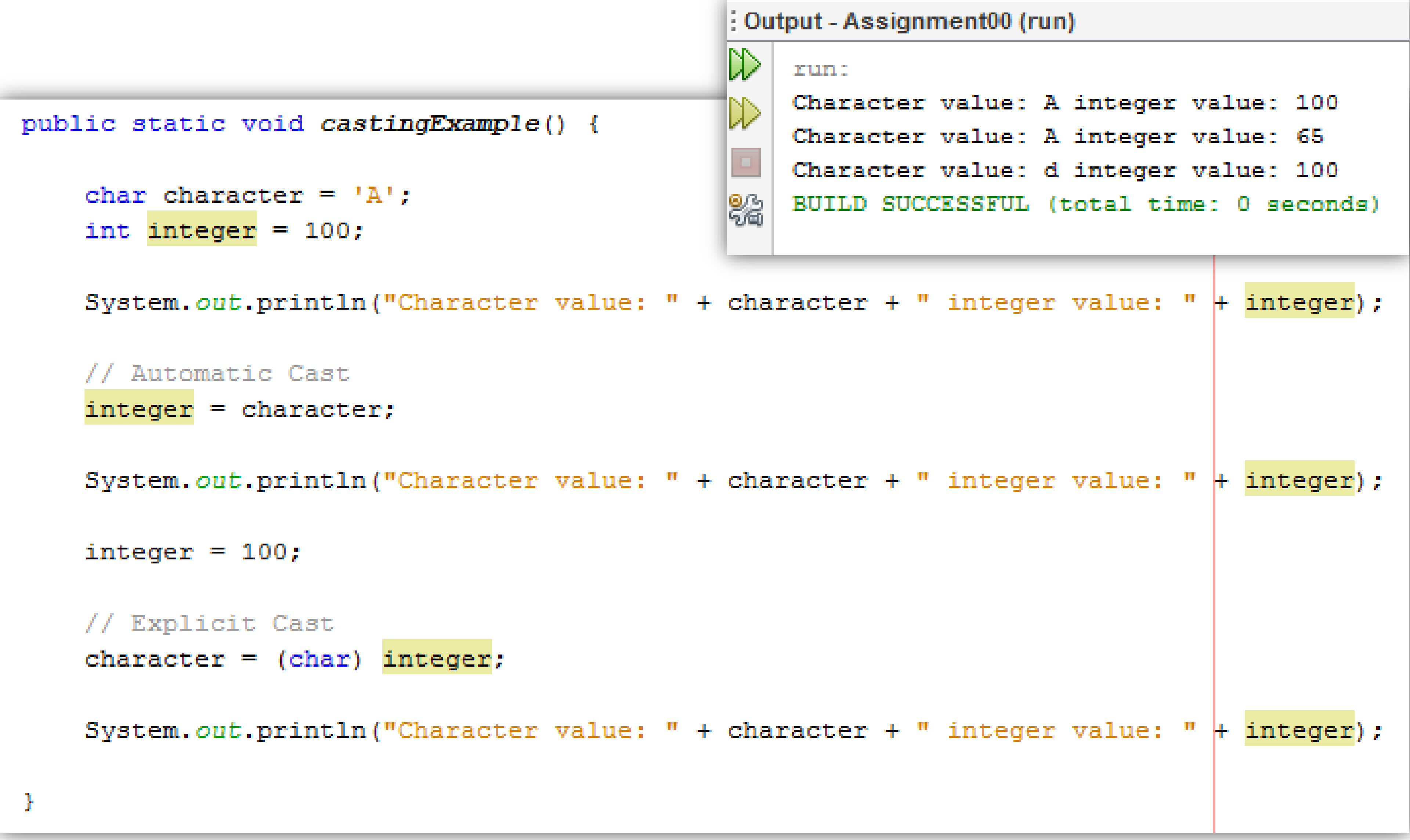
Primitive Types Casting

		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

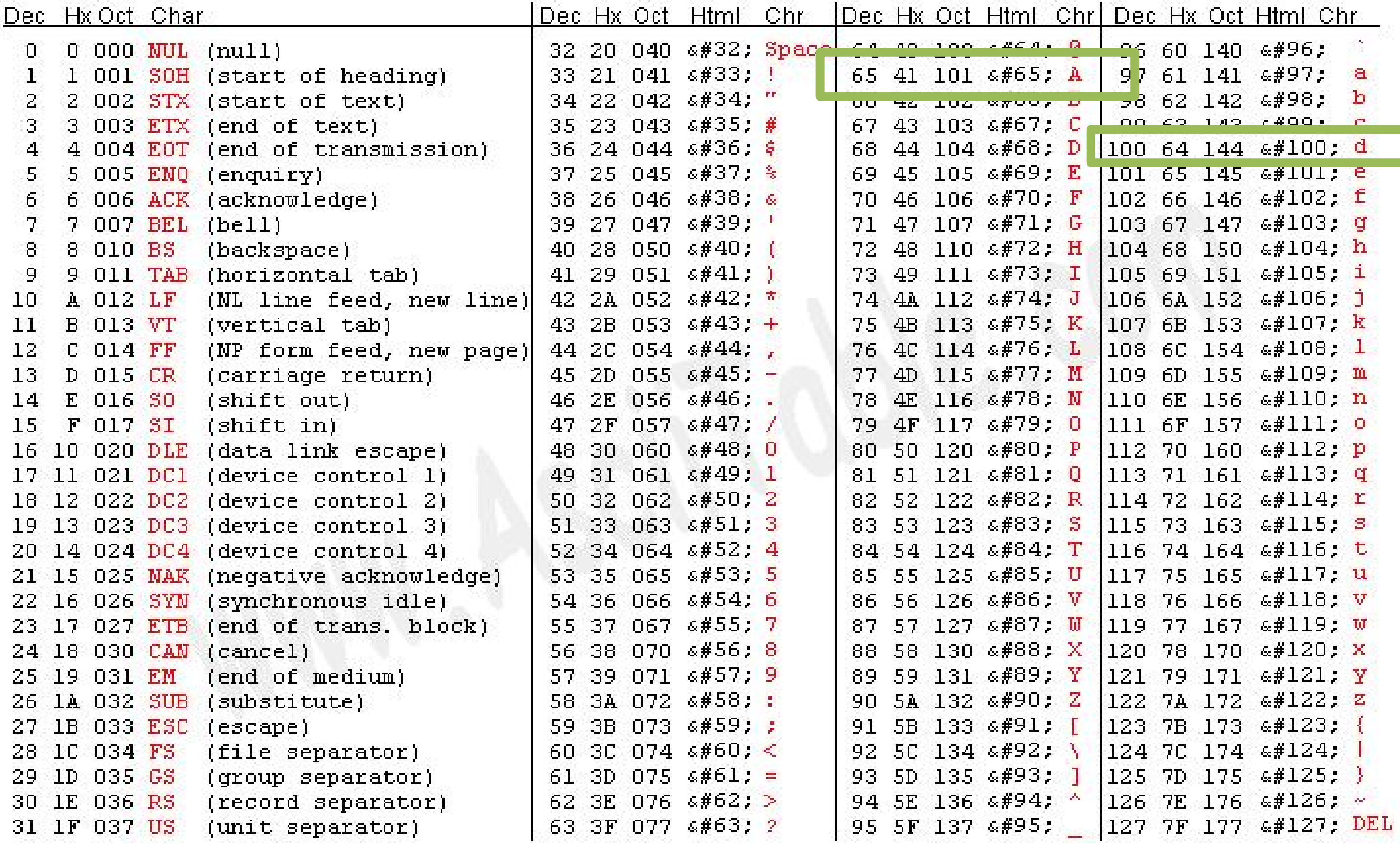


Output - Assignment00 (run)

```
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)
```

```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);

}
```

ASCII table



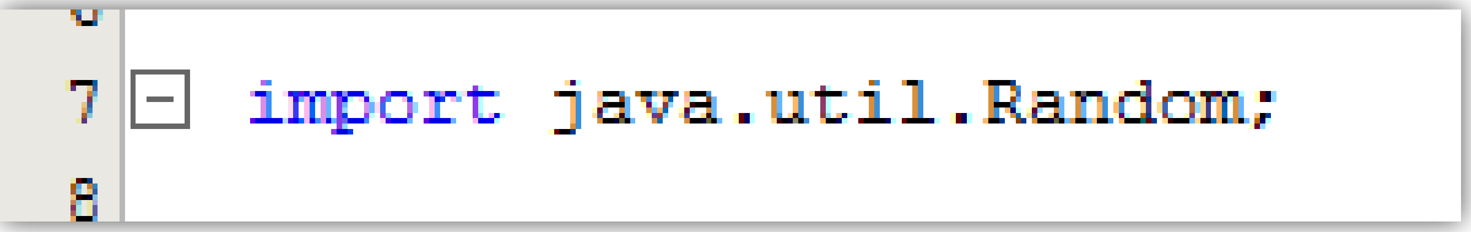
Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;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

Working with Random numbers

Generating random numbers

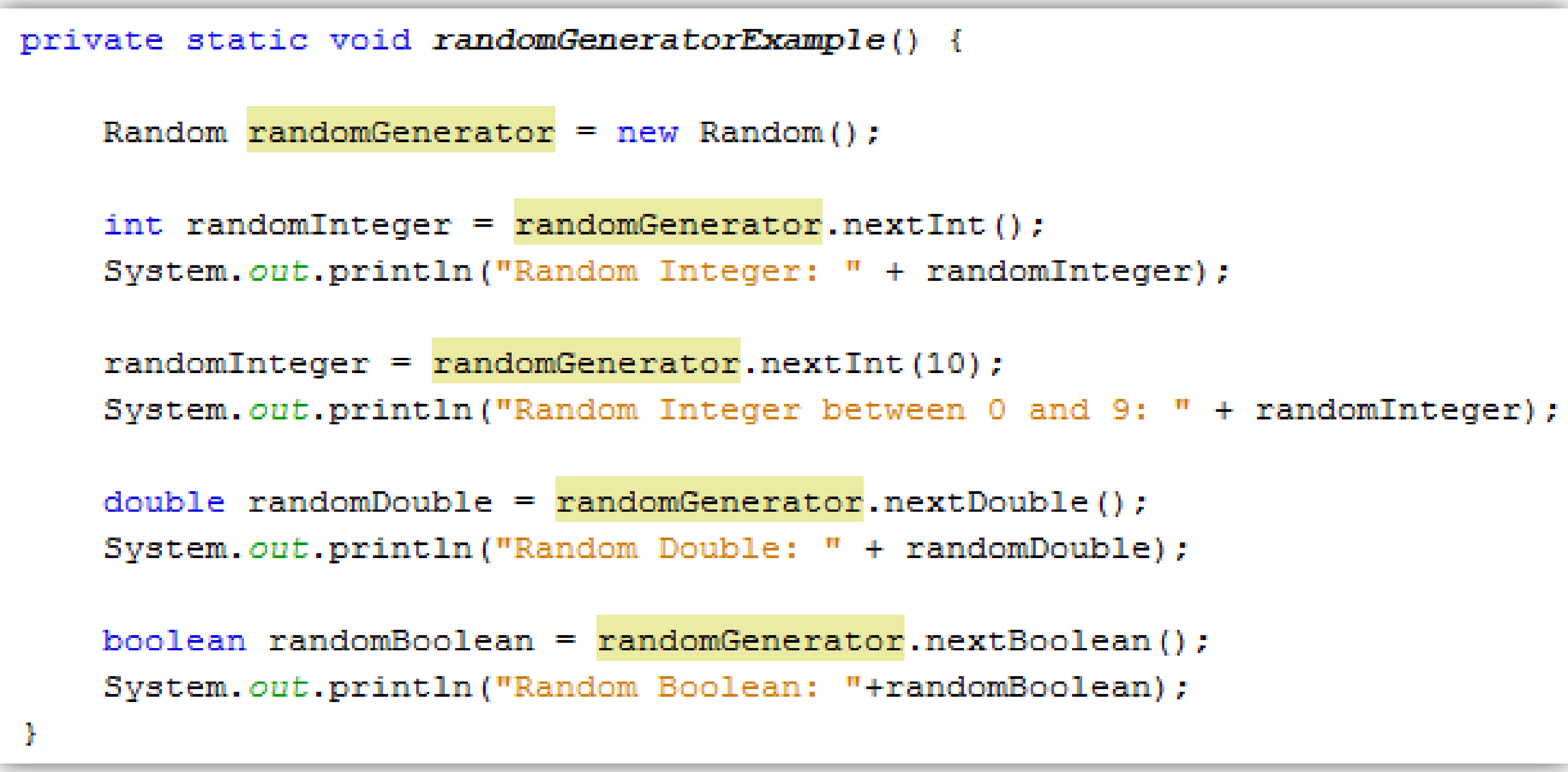
Remember to import the

Random Class



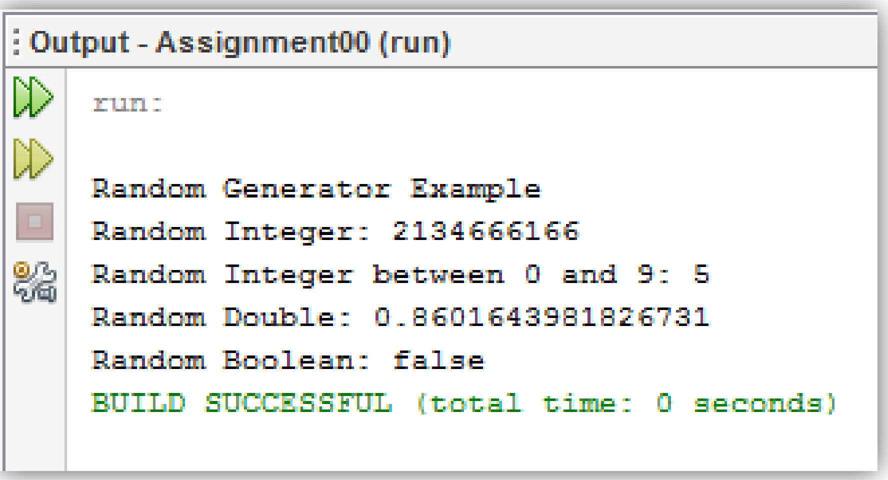
```java
import java.util.Random;
```

Generating random numbers

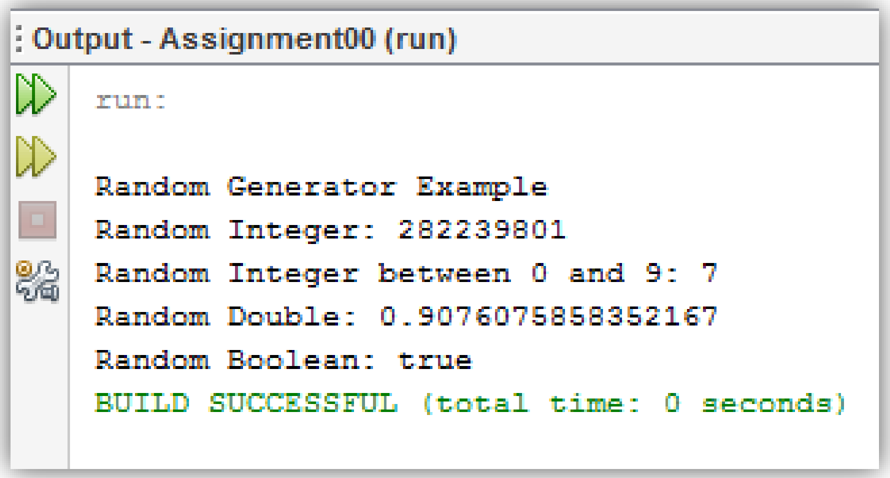


```java
private static void randomGeneratorExample() {

    Random randomGenerator = new Random();

    int randomInteger = randomGenerator.nextInt();
    System.out.println("Random Integer: " + randomInteger);

    randomInteger = randomGenerator.nextInt(10);
    System.out.println("Random Integer between 0 and 9: " + randomInteger);

    double randomDouble = randomGenerator.nextDouble();
    System.out.println("Random Double: " + randomDouble);

    boolean randomBoolean = randomGenerator.nextBoolean();
    System.out.println("Random Boolean: "+randomBoolean);
}
```

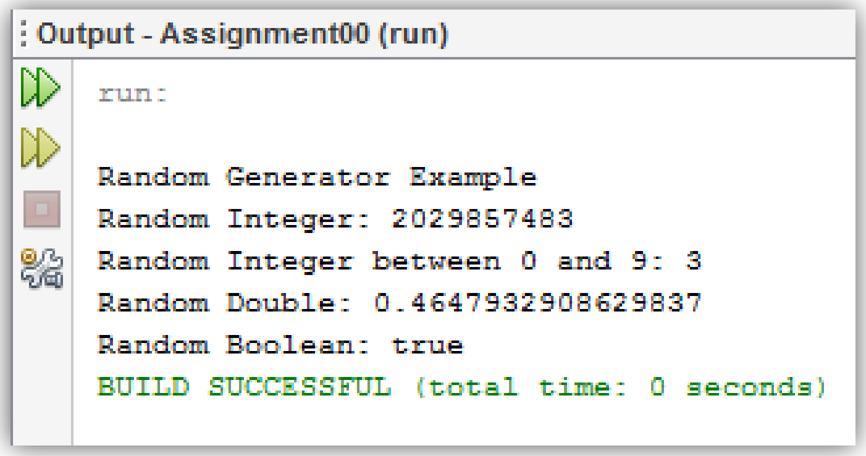
Generating random numbers



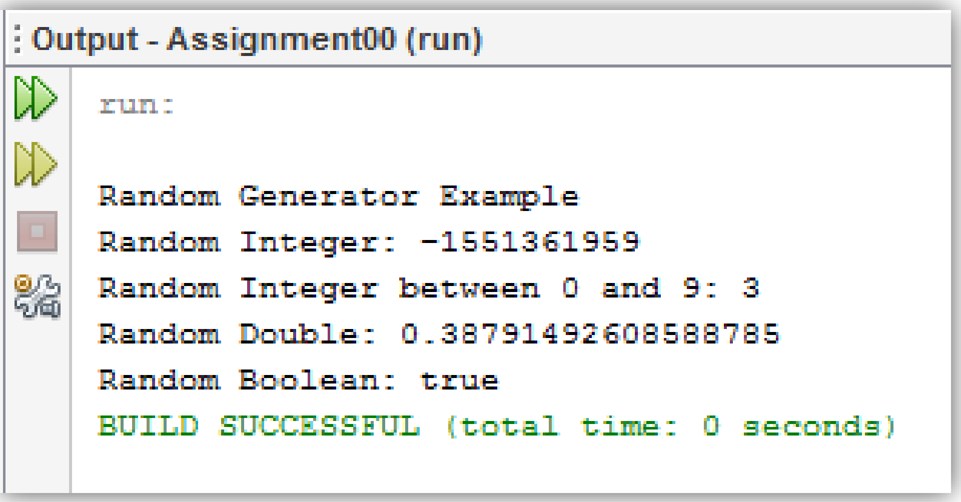
```
run:

Random Generator Example
Random Integer: 2134666166
Random Integer between 0 and 9: 5
Random Double: 0.8601643981826731
Random Boolean: false
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:

Random Generator Example
Random Integer: 282239801
Random Integer between 0 and 9: 7
Random Double: 0.9076075858352167
Random Boolean: true
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:

Random Generator Example
Random Integer: 2029857483
Random Integer between 0 and 9: 3
Random Double: 0.4647932908629837
Random Boolean: true
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:

Random Generator Example
Random Integer: -1551361959
Random Integer between 0 and 9: 3
Random Double: 0.3879149260858785
Random Boolean: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

HangMan

- Write the program HangMan
  - Program must choose randomly the secret word from a predefined list. (Given by me ☺)
  - Program must show after each player turn the current game state:
    - Which letters has been discovered. For example:
      - _ b _ e _ t _ (Objects)
    - How many errors has been committed until complete this figure

      q(x_x)p
    - Which letters has been used

Game Output example

System (Secret word = "object")	Player
_ _ _ _ _ _	
	User try the letter a
_ _ _ _ _ _ q a	
	User try the letter j
_ _ j _ _ _ q aj	
	User try the letter w
_ _ j _ _ _ q(ajw	
	User try the letter n
_ _ j _ _ _ q(X ajnw	
	User try the letter b
_ b j _ _ _ q(X abjnw	

Game Output example

System (Secret word = “object”)	Player
o b j e c _ q(X_X) abcde…	
	User try the letter t
o b j e c t q(X_X) abcde…	
You Win !!!	

System (Secret word = “object”)	Player
o b j e c _ q(X_X) abcde…	
	User try the letter p
o b j e c t q(X_X)P abcde…	
You Lose !!! The secret word is: object	

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.