Ο τύπος δεδομένων Χαρακτήρας και οι Δομές Ελέγχου For-Switch

Αθ. Ανδρούτσος

Χαρακτήρες

- Ένας χαρακτήρας (char) μπορεί να είναι οποιοδήποτε εκτυπώσιμο (ή μη-εκτυπώσιμο) σύμβολο από τα ακόλουθα:
  - **Γράμματα**, πεζά (lowercase) ή κεφαλαία (uppercase)
  - **Σημεία στίξης**, ?!~\|,. και άλλα
  - **Αριθμοί,** 0-9
  - **Μη-εκτυπώσιμοι** (non-printing) δηλαδή μη-ορατοί (non-visible) χαρακτήρες όπως space, carriage return, tab, esc, delete, κλπ.

Αναπαράσταση Χαρακτήρων

- Οι Η/Υ καταλαβαίνουν μόνο ακολουθίες από 1s και 0s
- Για να καταλάβουν χαρακτήρες πρέπει οι χαρακτήρες να μετατραπούν/ απεικονιστούν σε character codes ή code points, δηλαδή σε ακολουθίες από bits (bit-patterns), δηλαδή συνδυασμοί από 1 και 0
- Μία συλλογή χαρακτήρων και των αντίστοιχων code-points ονομάζεται **character set**

Bit-patterns

- Αν θα θέλαμε να μπορούμε να απεικονίσουμε δύο μόνο χαρακτήρες, για παράδειγμα το 'A' και 'B', τότε θα χρειαζόμασταν 1 bit.
- Θα μπορούσαμε να συμφωνήσουμε ότι το 0 θα είναι το A και το 1 θα είναι το B
- Αν θα θέλαμε να απεικονίσουμε 4 χαρακτήρες, θα χρειαζόμασταν 2 bits ώστε να μπορούμε να πάρουμε $2^2 = 4$ συνδυασμούς, (00, 01, 10, 11)
- Αν θα θέλαμε 8 χαρακτήρες, θα χρειαζόμασταν 3 bits, ώστε να πάρουμε $2^3 = 8$ συνδυασμούς (000, 001, 010, 011, 100, 101, 110, 111)
- Και ούτω καθεξής, αν θέλουμε περισσότερους χαρακτήρες

ASCII

- Το πρώτο επίσημο standard κωδικοποίησης χαρακτήρων ήταν το σύστημα **ASCII (American Standard Code for Information Interchange, 1963)** που χρησιμοποιούσε **7 bits** για να απεικονίσει 128 (0-127) λατινικούς χαρακτήρες, σημεία στίξης και non-printing characters
- Επειδή η μικρότερη δομική μονάδα αποθήκευσης πληροφοριών είναι το byte, **το 8ο bit κατά σύμβαση πλέον είναι το 0**, αν και αρχικά χρησιμοποιούνταν ως **parity bit** για τον έλεγχο λαθών κατά τη μετάδοση πληροφοριών

Πίνακας ASCII

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	SoH	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	SoTxt	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	EoTxt	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	EoT	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enq	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Ack	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Bsp	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	HTab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	LFeed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	VTab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	FFeed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	CR	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	SOut	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	SIn	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	DLE	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	DC1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	DC2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	DC3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	DC4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	NAck	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Syn	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	EoTB	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Can	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	EoM	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Sub	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Esc	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	FSep	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	GSep	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	RSep	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	USep	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Delete

- Βλέπετε για παράδειγμα πως το ordinal number (code point) του αγγλικού Α είναι το 65, του μικρού a το 97, και του χαρακτήρα 0 (μηδέν) το 48

ASCII – Μετατροπή bit-pattern στο δεκαδικό

- Ας υποθέσουμε τον χαρακτήρα Α, που αναπαρίσταται ως 0100 0001 (= 65 στο δεκαδικό)
- Η ακολουθία αυτή είναι μία δυαδική ακολουθία από 1 και 0 που μπορεί να αντιστοιχηθεί στο δεκαδικό σύστημα, αν βάλουμε βάρη σε κάθε θέση από δεξιά προς τα αριστερά

Τελική Τιμή $0*2^7 + 1*2^6 + 0*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 0*2^0 + 1*2^1 = 65$

Τιμή στο δεκαδικό σύστημα κάθε θέσης	128	64	32	16	8	4	2	1	= 65
Τιμή (δυνάμεις του 2) κάθε θέσης	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Αρίθμηση θέσεων (index) από δεξιά προς τα αριστερά	7	6	5	4	3	2	1	0	
Bit-Pattern	0	1	0	0	0	0	0	1	

ASCII Parity Bit

- Παλαιότερα το 8ο bit στο σύστημα ASCII ήταν **parity-bit** με άρτια ισοτιμία, αν δηλαδή είχαμε το Α που είναι 100 0001 τότε για να έχουμε άρτιο αριθμό από 1s το 8ο αριστερό bit θα ήταν το 0 (δηλ. **0**100 0001)
- Αν είχαμε το C που είναι το 67 = 100 0011, τότε για να έχουμε άρτιο αριθμό από 1s , το 8ο bit θα ήταν το 1 (δηλ. **1**100 0011)
- Κατ' αυτόν τον τρόπο το parity bit λειτουργούσε ως error check bit. Αν κατά τη μετάδοση άλλαζε ο αριθμός (λόγω error ή εσκεμμένα!) κατά περιττό αριθμό bit, θα μπορούσε να ελεγχθεί μιας και το parity θα επαναϋπολογιζόταν στον προορισμό και θα ήταν λάθος
- Τώρα πλέον όπως αναφέραμε, το 8ο bit στο ASCII είναι πάντα 0. Error detection κατά τη μετάδοση πληροφοριών γίνεται από τα πρωτόκολλα δικτύου με checksums, που βασίζονται σε αλγόριθμους κρυπτογράφησης

Unicode (1)

- Το πρόβλημα με το σύστημα ASCII είναι ότι απεικονίζει μόνο λατινικούς χαρακτήρες
- Υπάρχουν δύο τρόποι να επεκτείνουμε το σύστημα ASCII
- Ο 1ος τρόπος είναι να προσθέσουμε ένα 2ο byte και να έχουμε έτσι δύο bytes (16 bits), άρα να μπορούμε να απεικονίσουμε περισσότερους χαρακτήρες. Το πρόβλημα εδώ είναι ότι για να απεικονίσουμε τους ASCII χαρακτήρες, θα χρησιμοποιούμε και τα δύο bytes, ενώ θα αρκούσε μόνο το ένα. Το 2ο byte θα γεμίζει (padding) με μηδενικά. Άρα σπαταλούμε χώρο

Unicode (1)

- Επίσης, με **16-bits** μπορούμε να απεικονίσουμε **65536** χαρακτήρες, που είναι σχετικά λίγοι σε σχέση με τους χαρακτήρες όλων των γλωσσών (μόνο οι ιαπωνικοί είναι 80,000 χαρακτήρες)
- Ωστόσο, αυτό ακριβώς ήταν το 1ο standard που εφαρμόστηκε στην κωδικοποίηση χαρακτήρων στην Java με όνομα **UCS-2 (Universal Coded Character Set, ISO/IEC 10646) που είχε σταθερό μέγεθος 2 bytes**

UTF-16

- Το **UCS-2** επεκτάθηκε στη συνέχεια και δημιούργησε το **UTF-16** (16-bit Unicode Transformation Format) που είναι ένα μεταβλητού μεγέθους σύστημα (2-4 bytes)
- Το UTF-16 με μέγεθος 2-4 bytes μπορεί να απεικονίσει πάνω από 1.1 εκ. χαρακτήρες
- Το UTF-16 χρησιμοποιείται πλέον από την Java για την αναπαράσταση 1.114.112 χαρακτήρων

UTF-32

- Το **UTF-32** είναι παρόμοιο με το UTF-16 μόνο που έχει σταθερό μέγεθος 4 bytes και μπορεί να απεικονίσει 2^{32} χαρακτήρες
- Το μειονέκτημα είναι η σπατάλη χώρου μιας και συνήθως τα 11 bits είναι 0, εκτός από τις περιπτώσεις απεικόνισης ειδικών χαρακτήρων (emojis, κλπ.)

UTF-8 (1)

- Το UTF-16 και UTF-32 δεν χρησιμοποιούνται στην πράξη (π.χ. στο Web), χάριν ενός πιο αποδοτικού συστήματος κωδικοποίησης χαρακτήρων, του UTF-8
- **Το UTF-8 έχει μεταβλητό μέγεθος 1-4 bytes**
- Το 1ο byte όταν ξεκινάει από το 0, απεικονίζει τους 128 ASCII χαρακτήρες
- Η Java όπως αναφέραμε δεν χρησιμοποιεί UTF-8 αλλά UTF-16 για ιστορικούς λόγους σχεδίασης

UTF-8 (2)

- Στο UTF-8 όταν το 1ο byte ξεκινάει από το 110 χρησιμοποιούνται το 2ο και 3ο byte και όταν ξεκινάει από 11110 χρησιμοποιούνται και τα υπόλοιπα 3 bytes, άρα και το 4ο byte
- Το 4ο byte χρησιμοποιείται για την απεικόνιση CJK (Chinese, Japanese, Korean), μαθηματικών συμβόλων, emojis (expression/emotion symbols)

UTF-8 (3)

- Το σύστημα UTF-8 χρησιμοποιείται ευρέως στο encoding των χαρακτήρων στο Web
- Σχεδόν το 100% των Web Sites χρησιμοποιούν UTF-8
- Για παράδειγμα, το **$** αναπαρίσταται στο UTF-8 ως **U+0024** (εκφρασμένο στο δεκαεξαδικό σύστημα), χρησιμοποιεί 1 byte, με bit-pattern 0*010 0100* και στο δεκαεξαδικό σύστημα είναι το 24 (στο δεκαεξαδικό απλά αναπαριστούμε το bit-pattern στο δεκαδικό για κάθε κάθε 4-άδα bits, επομένως το 0010 είναι το 2 και το 0100 είναι το 4)

Τύπος Δεδομένων Char

- Η Java μας παρέχει τον **τύπο δεδομένων char** για να δηλώνουμε μεταβλητές τύπου χαρακτήρα
- Αποθηκεύονται ως απρόσημοι ακέραιοι 16-bit σύμφωνα με το σύστημα UCS-2 / UTF-16 μιας και τα δύο συστήματα είναι ίδια για το BMP (Basic Multilingual Plane, 65536 χαρακτήρες). Για χαρακτήρες πέρα από το BMP (supplementary characters) χρησιμοποιούνται δύο χαρακτήρες (surrogate pair)
- Σταθερές char (char literals) ορίζουμε μέσα σε **μονά ' '**

Java - Ο τύπος Χαρακτήρας (char)

- Οι **ακολουθίες διαφυγής (escape sequences)** είναι ακολουθίες χαρακτήρων (character sequences) που ξεκινούν με το **\ (backslash) και μεταφράζονται σε άλλους χαρακτήρες ή μη-εκτυπώσιμους χαρακτήρες**
- Αν θέλουμε να εμφανίσουμε μέσα σε println σημεία στίξης που χρησιμοποιούνται και από το συντακτικό της Java, όπως ' " \ αλλαγή γραμμής, tab τότε χρησιμοποιούμε τον παρακάτω συμβολισμό που περιλαμβάνει το backslash \ και τον χαρακτήρα που θέλουμε να εμφανίζουμε ή τον non-printing char

\ ' (το ίδιο το ') **\ "** (το ίδιο το ") **** (το ίδιο το \)
\n (new line) **\t** (tab)

Δηλώσεις μεταβλητών και σταθερών τύπου Χαρακτήρα


```
1   package gr.aueb.cf.ch4;
2
3   /**
4    * Char data type
5    */
6   public class DeclarationApp {
7
8       public static void main(String[] args) {
9           final char EMPTY_CHAR = ' ';
10          char star = '*';
11          char chA = 'A';
12      }
13  }
```

- Μεταβλητές τύπου char δηλώνουμε ως **char** Σταθερές char δηλώνουμε με **final**. Char literals ορίζουμε μέσα σε ' ' (single quotes)

Hello με chars


```java
package gr.aueb.cf.ch4;

/**
 * Char Hello App
 */
public class CharHelloApp {

    public static void main(String[] args) {
        char h = 'H';
        char e = 'e';
        char l = 'l';
        char o = 'o';
        char exclMark = '!';

        System.out.print(h);
        System.out.print(e);
        System.out.print(l);
        System.out.print(l);
        System.out.print(o);
        System.out.print(exclMark);
    }
}
```

- Δηλώνουμε και αρχικοποιούμε μεταβλητές τύπου char με διάφορους χαρακτήρες ώστε να σχηματίσουμε τη λέξη Hello!

Συγκρίσεις

- Συγκρίσεις κάνουμε με σχεσιακούς τελεστές
  - `'A'<'B', '2'<'3', 'z' > 'a'` όλα true
  - `'9'<'8', 'b'<'a', 'z' != 'z'` όλα false
- Οι παραπάνω συγκρίσεις εκτελούνται αφού πρώτα οι σταθερές char μετατρέπονται σε int σύμφωνα με το UCS-2 / UTF-16 character set

Συγκρίσεις char


```java
package gr.aueb.cf.ch4;

/**
 * Chars' collation/sorting is based on code points.
 * That is comparisons between chars is based on the
 * ordinal numbers (code points) according to UCS-2
 * (UTF-16). UCS-2 includes ASCII characters as well as
 * Greek characters in two-byte sequences (bit patterns).
 */
public class CompareApp {

    public static void main(String[] args) {
        char a = 'α';   // Ελληνικό α
        char b = 'b';   // Αγγλικό b
        boolean greekIsGreater = false;

        System.out.println((int) a);
        System.out.println((int) b);

        if (a > b) greekIsGreater = true;

        System.out.println( "Greek is greater: " + greekIsGreater);
    }
}
```

```
Run:  CompareApp
"C:\Program Files\Amazon
945
98
Greek is greater: true
```

- Οι ελληνικοί χαρακτήρες έχουν μεγαλύτερο ordinal number από τα Latin
- Επίσης τα κεφαλαία έχουν πιο μικρό ordinal και στα Latin και στα Ελληνικά

Έξοδος τιμών τύπου char


```java
package gr.aueb.cf.ch4;

/**
 * char output.
 */
public class CharPrintApp {

    public static void main(String[] args) {
        char ch1 = 's';
        char ch2 = '8';

        System.out.println("char: " + ch1 + ", ordinal: " + (int) ch1);
        System.out.println("char: " + ch2 + ", ordinal: " + (int) ch2);

        System.out.printf("char: %c , ordinal: %d\n", ch1, (int) ch1);
        System.out.printf("char: %c , ordinal: %d\n", ch2, (int) ch2);
    }
}
```

- Με typecast σε (int) εκτυπώνουμε το ordinal value
- Στην printf με %c εκτυπώνουμε chars και με %d ordinal number (είναι int)

Surrogate pairs


```java
package gr.aueb.cf.ch4;

/**
 * Unicodes greater than 4 Hex digits
 * could be represented in UTF-16 by
 * surrogate pairs.
 * There are surrogate pairs calculators.
 * Or by calling Character wrapper class with
 * toString method.
 */
public class UnicodeApp {

    public static void main(String[] args) {
        int codePoint = 0x1F600;    // smiley

        // conversion of 0x1F600 to Surrogate Pairs
        // http://russellcottrell.com/greek/utilities/SurrogatePairCalculator.htm
        System.out.println("Smiley: \uD83D\uDE00");

        // Java-based conversion with Character wrapper class
        // and toChars method that converts to UTF-16 code points
        System.out.print("Smiley: ");
        System.out.println(Character.toChars(codePoint));
    }
}
```

Τα UTF-8 code points που είναι μεγαλύτερα από 2 bytes για να μετατραπούν σε UTF-16 θα πρέπει να μετατραπούν σε surrogate pairs, που είναι ακολουθίες δύο UTF-16 code points που διευρύνουν το φάσμα των 65536 χαρακτήρων σε πάνω από 1.1 εκ.

Μπορούμε να μετατρέψουμε από UTF-8 σε UTF-16 surrogate pairs

http://russellcottrell.com/greek/utilities/SurrogatePairCalculator.htm

Emojis (1)

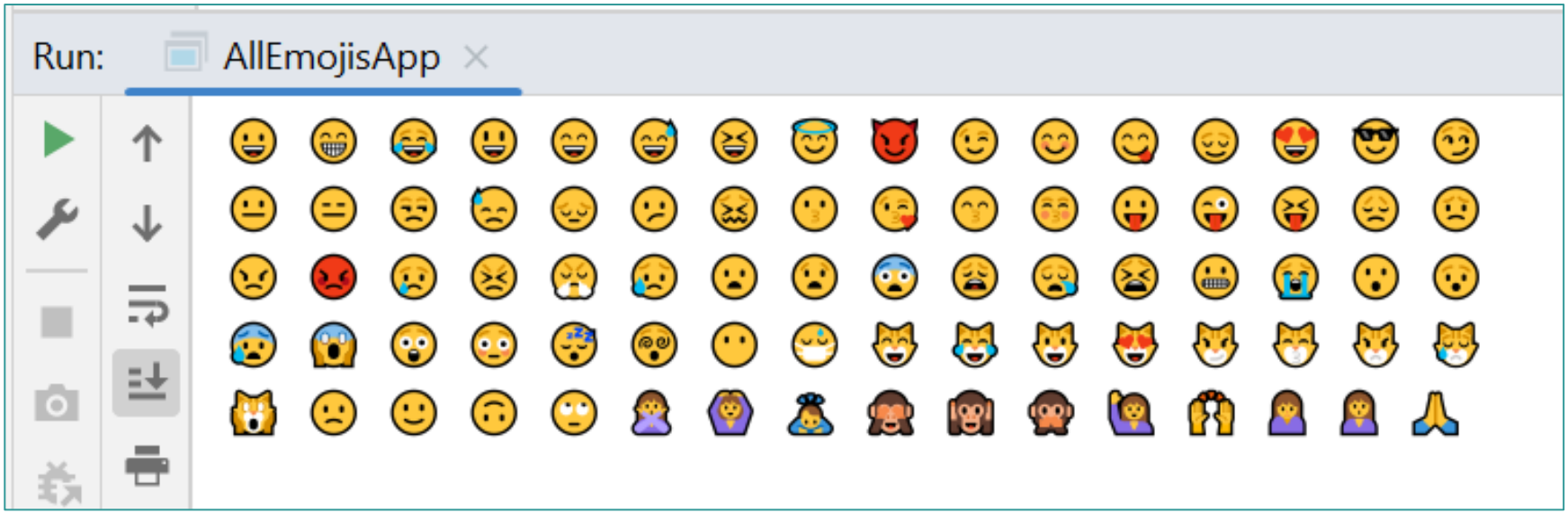

```java
package gr.aueb.cf.ch4;

/**
 * Εκτυπώνει όλα τα emojis στο range  0x1F600 - 0x1F64F.
 Η Character.toChars(UnicodeCodePoint), αν το UnicodeCodePoint
 * ανήκει στο Plane 0, (δηλαδή U+0000 - U+FFFF) επιστρέφει το ίδιο
 * το code point, αλλιώς αν το UnicodeCodePoint είναι supplementary
 * code point (ανήκει σε μεγαλύτερο Plane (1-16) και έχει επομένως
 * 5 Hex numbers, π.χ. 0x1F600 ) επιστρέφει το surrogate pair.
 * Στο σύστημα Unicode τα Planes είναι συνεχόμενα groups από 65536 (2^16)
 * χαρακτήρες.
 */
public static void main(String[] args) {
    int emojiStart = 0x1F600;
    int emojiEnd = 0x1F64F;
    int counter = 0;
    int emoji;

```

Emojis (2)


```
15  public static void main(String[] args) {
16      int emojiStart = 0x1F600;
17      int emojiEnd = 0x1F64F;
18      int counter = 0;
19      int emoji;
20
21      emoji = emojiStart;
22      while (emoji < emojiEnd) {
23          System.out.print(Character.toChars(emoji));
24          System.out.print(" ");
25          emoji++;
26          counter++;
27          if (counter % 16 == 0) {
28              System.out.println();
29          }
        }
    }
```

- Με counter % 16 κάθε δεκαέξι στοιχεία κάνουμε println

Αποτέλεσμα



Είσοδος τιμών τύπου char (1)


```java
package gr.aueb.cf.ch4;

import java.io.IOException;

/**
 * Char input with System.in.read()
 */
public class CharInputApp {

    public static void main(String[] args) throws IOException {
        int inputChar = ' ';    // ' ' returns the ordinal value of the char

        System.out.println("Please insert an ASCII char");
        inputChar = System.in.read();   // reads one byte as int

        System.out.println("char: " + (char) inputChar);
    }
}
```

- Η *System.in.read*() διαβάζει ένα byte τη φορά, οπότε διαβάζει σωστά μόνο ASCII, όχι UTF-16. Επίσης, επειδή είναι χαμηλού επιπέδου μηχανισμός θεωρεί ότι μπορεί να συμβεί κάποιο λάθος κατά την ανάγνωση και για αυτό απαιτεί η μέθοδος main να κάνει throws *IOException*

Είσοδος τιμών τύπου char (2)


```java
package gr.aueb.cf.ch4;

import java.util.Scanner;

/**
 * Διαβάζει char με Scanner.
 */
public class CharScannerApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        char inputChar = ' ';

        // Η nextLine επιστρέφει όλη τη γραμμή μέχρι το \n
        // Η charAt(0) επιστρέφει τον πρώτο char ως UTF-16
        inputChar = in.nextLine().charAt(0);

        System.out.println("Input char: " + inputChar);
    }
}
```

- Μπορούμε να διαβάσουμε και με Scanner όλο το input μέχρι την αλλαγή γραμμής και μετά να εκχωρήσουμε με charAt(0) τον 1ο χαρακτήρα (που είναι στη θέση 0)

Η εντολή επανάληψης for

- Οι **εντολές while .. do και do .. while** παρέχουν συγκεκριμένο πλήθος επαναλήψεων αλλά για αυτό χρειάζεται μία μεταβλητή στο σώμα τους που ελέγχει τον τερματισμό της συνθήκης ελέγχου
- Η **εντολή for** παρέχει ***συγκεκριμένο πλήθος επαναλήψεων*** δίχως να χρειάζεται στο σώμα της μεταβλητή ελέγχου

for: σύνταξη και παραδείγματα

- for (Αρχική_τιμή; Έλεγχος; βήμα)
  - *for (int i = 1; i <= 5; i++)*
    //εκτελείται 5 φορές

- Η ***δήλωση της i*** (που είναι η μεταβλητή ελέγχου) μπορεί να γίνει μέσα στη for και τότε η εμβέλεια της i είναι μόνο μέσα στη for (τοπική μεταβλητή της for)
- Εμβέλεια σημαίνει πως **έξω από τη for δεν υπάρχει i**
- Κάτι τέτοιο είναι πιο αποδοτικό σε σχέση με τη while όπου το i είχε δηλωθεί έξω από την while και υπήρχε και μετά το τέλος της while

For - Παράδειγμα


```java
package gr.aueb.cf.ch4;

/**
 * Prints each iteration-number.
 */
public class For1App {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.print(i + " ");
        }
    }
}
```

- Παρατηρούμε πως εκτελείται 10 φορές, αφού η αρχική τιμή του i είναι 1, η τελική τιμή του i είναι 10 και το βήμα είναι 1

Σε κάθε iteration εκτυπώνει το i καθώς και ένα κενό διάστημα (για να υπάρχει ένα κενό διαχωριστικό μεταξύ των αριθμών που εκτυπώνονται)

Άθροισμα/Γινόμενο 10 πρώτων αριθμών


```java
package gr.aueb.cf.ch4;

/**
 * Υπολογίζει και εκτυπώνει το άθροισμα
 * και το γινόμενο των 10 πρώτων αριθμών.
 */
public class SumMulApp {

    public static void main(String[] args) {
        int sum = 0;
        int mul = 1;

        for (int i = 1; i <= 10; i++) {
            sum += i;
            mul *= i;
        }

        System.out.println("Sum is: " + sum);
        System.out.println("Mul is: " + mul);
    }
}
```

- Για το άθροισμα αρχικοποιούμε το αποτέλεσμα, στη μεταβλητή sum, στο 0
- Για γινόμενο αρχικοποιούμε την mul στο 1
- Η for εκτελείται από i = 1 έως i = 10 με βήμα i++ , άρα εκτελείται 10 φορές.
- Σε κάθε επανάληψη προσθέτουμε το i στο sum και πολλαπλασιάζουμε επί i στο mul

Δύναμη του a εις την b


```java
import java.util.Scanner;

/**
 * Υπολογίζει το a^b, π.χ. το 2^3 = 8.
 * Διαβάζει τα a και b από τον χρήστη.
 */
public class PowerApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = 0;
        int b = 0;
        int result= 1;

        System.out.println("Please insert a, b (ints)");
        a = in.nextInt();
        b = in.nextInt();

        for (int i = 1; i <= b; i++) {
            result = result * a;
        }

        System.out.printf("%d^%d = %d", a, b, result);
    }
}
```

- Αφού κατανοήσουμε ότι το a^b είναι ένας πολλαπλασιασμός του a*a*..*a, b φορές, μπορούμε να ορίσουμε μία for, από i = 1 έως i <= b, με βήμα i++
- Το result αρχικοποιείται στο 1 ως ουδέτερο στοιχείο του πολλαπλασιασμού και μέσα στη for γίνεται σε κάθε επανάληψη result = result * a

Big Integers


```java
import java.math.BigInteger;
import java.util.Scanner;

/**
 * Υπολογίζει το a^b για Big Integers.
 */
public class BigIntApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        BigInteger a = BigInteger.ZERO;
        BigInteger b = BigInteger.ZERO;
        BigInteger result = BigInteger.ONE;

        System.out.println("Please insert two ints");
        a = BigInteger.valueOf(in.nextInt());
        b = BigInteger.valueOf(in.nextInt());

        for (int i = 1; i <= b.intValue(); i++) {
            result = result.multiply(a);
        }

        System.out.printf("%d^%d = %,d", a, b, result);
    }
}
```

- Με BigInteger.valueOf(int) μετατρέπουμε από int σε Big Integer
- Με bigInt.intValue() μετατρέπουμε μία μεταβλητή bigint σε int
- Με bigInt.multiply(bigInt) πολλαπλασιάζουμε δύο big integers
- Εκτυπώνουμε με %d όπως τους ints

Flexible for


```java
/**
 * Flexible for. Διαβάζει από το stdin την
 * αρχική τιμή, τελική τιμή και το step.
 * Εκτυπώνει κάθε τιμή του i και το πλήθος
 * των επαναλήψεων.
 */
public class ForFlexApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int startValue = 0;
        int endValue = 0;
        int step = 0;
        int iterations = 0;

        System.out.println("Please insert start, end, step (ints)");
        startValue = in.nextInt();
        endValue = in.nextInt();
        step = in.nextInt();

        for (int i = startValue; i <= endValue; i = i + step) {
            iterations++;
            System.out.print(i + " ");
        }

        System.out.println();
        System.out.println("Iterations: " + iterations);
    }
}
```

- Διαβάζει από τον χρήστη, την αρχική τιμή του i, την τελική τιμή και το βήμα αύξησης
- Υπολογίζει τις επαναλήψεις (iterations) και εκτυπώνει την τιμή του i σε κάθε επανάληψη

break


```java
package gr.aueb.cf.ch4;

/**
 * Break stops execution.
 */
public class ForBreakApp {

    public static void main(String[] args) {

        for (int i = 1; i <= 10; i++) {
            System.out.print(i + " ");
            if (i == 5) break;
        }

        System.out.println();
        System.out.println("for loop stopped...");
    }
}
```

- Η εντολή break διακόπτει την for (όπως διακόπτει και την while)
- Στο παράδειγμα διακόπτει όταν το i είναι 5

Αέναο for


```java
package gr.aueb.cf.ch4;

public class ForEverApp {

    public static void main(String[] args) {
        int count = 0;

        for (;;) {
            System.out.print("forever");
            count++;
            if (count % 20 == 0) System.out.println();
            if (count == 100) break;
        }
    }
}
```

- Αέναο for loop με ;;
- Όταν το count φτάσει στο 100 κάνουμε break αλλιώς θα τρέχει για πάντα

Η εντολή Switch

- Χρησιμοποιείται για να μην έχουμε πολλά if-then-else
- switch (obj) {
  case:
  εντολές; break;
  default:
  εντολή; break;
  }

  όπου *obj* μπορεί να είναι: int, byte, short, char, String, enum, Integer, Byte, Short, Character
- Το **default** εκτελείται όταν δεν εκτελεστεί κανένα case

Switch – Παράδειγμα (1)


```
5   /**
6    * Switch instead of many if-then-else.
7    * Gets user's choice and gives feedback.
8    */
9   public class SwitchApp {
10
11      public static void main(String[] args) {
12          Scanner in = new Scanner(System.in);
13          int choice = 0;
14
15          System.out.println("Please select one of the following");
16          System.out.println("1. One-player game");
17          System.out.println("2. Two-player game");
18          System.out.println("3. Team game");
19          System.out.println("4. Exit");
20          System.out.println("Please insert your choice");
21
22          // Get the choice
23          choice = in.nextInt();
```

- Εμφανίζει το μενού και διαβάζει την επιλογή του χρήστη
- Στη συνέχεια (στην επόμενη διαφάνεια) ελέγχει την επιλογή και δίνει feedback

Switch – Παράδειγμα (2)


```
25          switch (choice) {
26              case 1:
27                  System.out.println("Start one-player game");
28                  break;
29              case 2:
30                  System.out.println("Start two-player game");
31                  break;
32              case 3:
33                  System.out.println("Start team game");
34                  break;
35              case 4:
36                  System.out.println("Exit game");
37                  break;
38              default:
39                  System.out.println("Error in choice");
40                  break;
41          }
42      }
43  }
```

- Ελέγχει την τιμή της choice και ανάλογα με το αν είναι 1, 2, 3, 4 εκτελείται το αντίστοιχο case
- Αν η τιμή της choice είναι διάφορη από 1, 2, 3, 4, τότε εκτελείται το default
- Η break είναι απαραίτητη, διαφορετικά συνεχίζει να εκτελείται και η από κάτω case

Switch με do-while (1)


```java
/**
 * Gets user's choice and gives feedback,
 * inside a do - while.
 */
public class SwitchApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int choice = 0;

        do {
            System.out.println("Please select one of the following");
            System.out.println("1. One-player game");
            System.out.println("2. Two-player game");
            System.out.println("3. Team game");
            System.out.println("4. Exit");
            System.out.println("Please insert your choice");

            // Get the choice
            choice = in.nextInt();
```

- Όπως πριν, μόνο που τώρα η switch τρέχει επαναληπτικά μέσα σε μία do-while μέχρι το choice να γίνει 4

Switch με do-while (2)


```
26              switch (choice) {
27                  case 1:
28                      System.out.println("Start one-player game");
29                      break;
30                  case 2:
31                      System.out.println("Start two-player game");
32                      break;
33                  case 3:
34                      System.out.println("Start team game");
35                      break;
36                  case 4:
37                      System.out.println("Exit game");
38                      break;
39                  default:
40                      System.out.println("Error in choice");
41                      break;
42              }
43          } while (choice != 4);
44
45          System.out.println("Goodbye!");
46      }
47  }
```

- Η switch εκτελείται επαναληπτικά μέχρι το choice να γίνει 4

Υπολογιστής τσέπης (1)

- Θα υλοποιήσουμε ένα απλό υπολογιστή τσέπης που κάνει πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση και mod (υπόλοιπο)
- Θα δίνει ο χρήστης την πράξη, π.χ. 2 + 5 ή 12 / 4 ή 89 % 5 και ο υπολογιστής ανάλογα με το σύμβολο της πράξης θα κάνει τον αντίστοιχο υπολογισμό και θα εμφανίζει το αποτέλεσμα

Υπολογιστής τσέπης (2)


```
5   /**
6    * Διαβάζει από τον χρήστη ένα αριθμό (ακέραιο),
7    * ένα σύμβολο πράξης, και ένα ακόμα ακέραιο και
8    * εκτελεί την πράξη, ανάλογα με το σύμβολο που
9    * έχει δοθεί: +, -, *, /, %
10   */
11  public class CalculatorApp {
12
13      public static void main(String[] args) {
14          Scanner in = new Scanner(System.in);
15          int num1 = 0;
16          int num2 = 0;
17          int result = 0;
18          int choice = 0;
19          char operator = ' ';
20          boolean isError = false;
21
22          System.out.println("Please insert an int, an operator and a second int");
23          num1 = in.nextInt();
24          operator = in.next().charAt(0);
25          num2 = in.nextInt();
```

- Διαβάζουμε τον 1ο αριθμό, μετά το σύμβολο της πράξης και μετά τον 2ο αριθμό
- Στην επόμενη διαφάνεια ακολουθεί η switch

Υπολογιστής τσέπης (3)


```
27          switch (operator) {
28              case '+':
29                  result = num1 + num2;
30                  break;
31              case '-':
32                  result = num1 - num2;
33                  break;
34              case '*':
35                  result = num1 * num2;
36                  break;
37              case '/':
38                  if (num2 != 0) {
39                      result = num1 / num2;
40                  }
41                  break;
42              case '%':
43                  if (num2 != 0) {
44                      result = num1 % num2;
45                  }
46                  break;
47              default:
48                  System.out.println("Error in operator");
49                  isError = true;
50                  break;
51          }
```

```
53          if (!isError)
54              System.out.printf("%d %c %d = %d", num1, operator, num2, result);
55      }
56  }
```

- Σε κάθε case κάνουμε την αντίστοιχη πράξη. Κάθε case πρέπει να έχει ένα break αλλιώς η switch προχωράει και παρακάτω
- Το default πρέπει να υπάρχει για να καλύπτει όλες τις άλλες περιπτώσεις
- Υπάρχει και μία boolean *isError* που ενημερώνεται στο default ώστε να μην εμφανίζει η println σε περίπτωση λάθους

Switch fall-through (1)


```
1   package gr.aueb.than.ch4;
2
3   import java.util.Scanner;
4
5   /**
6    * Λαμβάνει ένα αριθμό που συμβολίζει βαθμό,
7    * από τον χρήστη και εμφανίζει κατάλληλο μήνυμα.
8    *
9    * @author A. Androutsos
10   */
11  public class SwitchFallThrough {
12
13      public static void main(String[] args) {
14          Scanner in = new Scanner(System.in);
15          int choice;
```

- Όταν δεν έχουμε break στα cases της switch, η εκτέλεση προχωράει στο επόμενο case
- Αυτή η ιδιότητα ονομάζεται ***fall-through***

- Δείτε την επόμενη διαφάνεια …

Switch fall-through (2)


```
16          choice = in.nextInt();
17
18          // switch that falls-through
19          switch (choice) {
20              case 1:
21              case 2:
22              case 3:
23              case 4:
24                  System.out.println("Κάτω από τη βάση");
25                  break;
26              case 5:
27              case 6:
28                  System.out.println("Καλώς");
29                  break;
30              case 7:
31              case 8:
32                  System.out.println("Λίαν Καλώς");
33                  break;
34              case 9:
35              case 10:
36                  System.out.println("Άριστα");
37                  break;
38              default:
39                  System.out.println("Παρακαλώ δώστε αριθμό 1 - 10");
40                  break;
41          }
42      }
43  }
```

- Το fall-through είναι σαν να έχω OR
- Δεν μπορούμε να γράψουμε case 1 || case 2 κλπ.

Παραδείγματα

- Εμφανίστε 10 αστεράκια (οριζόντια και κάθετα)
- Εμφανίστε 10 σειρές με 10 αστεράκια
- Εμφανίστε 10 σειρές: η $1^{η}$ σειρά με 1 αστεράκι, η $2^{η}$ σειρά 2 αστεράκια κλπ μέχρι την $10^{η}$ με 10 αστεράκια
- Εμφανίστε 10 σειρές: η $1^{η}$ σειρά 10 αστεράκια, η $2^{η}$ σειρά 9 αστεράκια κλπ μέχρι την $10^{η}$ με 1 αστεράκι

Εμφάνισε 10 αστεράκια

- *Εμφανίστε 10 αστεράκια οριζόντια*
- *Εμφανίστε 10 αστεράκια κάθετα*
- Σκεφτόμαστε ότι για να εμφανίσουμε 10 αστεράκια αρκεί να χρησιμοποιήσουμε τη δομή ελέγχου **for εμφανίζοντας 1 αστεράκι 10 φορές**
- Στον προγραμματισμό τα περισσότερα πράγματα γίνονται επαναληπτικά!

Εμφάνισε 10 αστεράκια οριζόντια


```java
package gr.aueb.cf.chapter4;

/**
 * Εκτυπώνει 10 οριζόντια αστεράκια
 */
public class StarsHorizontalDemo {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.print("*");
        }
    }
}
```

- Θέλουμε τα αστεράκια να εκτυπωθούν οριζόντια
- Γιαυτό, χρησιμοποιούμε την print και όχι println γιατί δεν θέλουμε αλλαγές γραμμής σε κάθε αστεράκι

Εμφάνισε 10 αστεράκια κάθετα


```java
package gr.aueb.cf.chapter4;

/**
 * Εκτυπώνει 10 κάθετα αστεράκια
 */
public class StarsVerticalDemo {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println("*");
        }
    }
}
```

- Θέλουμε τα αστεράκια να εκτυπωθούν κάθετα
- Γιαυτό, χρησιμοποιούμε την println() γιατί ***θέλουμε αλλαγές γραμμής*** σε κάθε αστεράκι

Εμφάνισε 10 σειρές με 10 αστεράκια

- Γράψτε ένα πρόγραμμα που να εμφανίζει 10 σειρές με 10 αστεράκια σε κάθε σειρά
- Σκεφτόμαστε ότι για να εμφανίσουμε 10 σειρές με 10 αστεράκια αρκεί να χρησιμοποιήσουμε **δύο επαναλήψεις η μία μέσα στην άλλη**
- Η **1η επανάληψη θα ελέγχει τις 10 σειρές** και η 2η εμφωλιασμένη(nested) επανάληψη για κάθε σειρά (από τις 10) θα εμφανίζει 10 αστεράκια

Εμφάνισε 10 σειρές με 10 αστεράκια


```java
package gr.aueb.cf.ch4;

/**
 * Εμφανίζει 10x10 αστεράκια.
 */
public class StarsHV10 {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            for (int j = 1; j <= 10; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

- Στο εξωτερικό for που τρέχει 10 φορές υπάρχουν **δύο πράξεις**:
- **(i)** ένα εσωτερικό for που εμφανίζει 10 αστεράκια οριζόντια και
- **(ii)** ένα println() για μία αλλαγή γραμμής μετά από κάθε γραμμή με 10 αστεράκια

Εμφάνισε 1, 2, 3,…, 10 αστεράκια (1)

- Εκτυπώστε 10 γραμμές όπου στην 1η γραμμή θα εκτυπώνει 1 αστεράκι, στην 2η γραμμή 2 αστεράκια, στην 3η γραμμή 3 αστεράκια κοκ μέχρι την 10η γραμμή όπου θα εκτυπώνει 10 αστεράκια
- Σκεφτόμαστε ότι για να εμφανίσουμε 10 σειρές χρειαζόμαστε ένα εξωτερικό for για τις 10 σειρές
- Σκεφτόμαστε ότι χρειαζόμαστε ένα ***εσωτερικό for που να ελέγχεται από το εξωτερικό*** που να εκτυπώνει 1 αστεράκι στην 1η σειρά , 2 στην 2η , κλπ. μέχρι 10 αστεράκια στην 10η σειρά

Εμφάνισε 1, 2, 3,…, 10 αστεράκια (2)


```java
package gr.aueb.cf.ch4;

/**
 * Εμφανίζει 1ο σειρές από αστεράκια
 * σε αύξουσα σειρά, 1 αστεράκι στη
 * 1η σειρά, 2 στη 2η σειρά, κοκ,
 * 10 αστεράκια στη 10η σειρά.
 */
public class StarsAsc10 {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

- Όταν το εξωτερικό for θα έχει τιμή 1 (1η σειρά) και το αστεράκι θα είναι 1
- Όταν το εξωτερικό for θα έχει τιμή 2 (2η γραμμή) τα αστεράκια θα είναι 2, κλπ., μέχρι τη 10η γραμμή όπου η τιμή του i στην εξωτερική for θα είναι 10 και τότε και τα αστεράκια θα είναι 10
- Οπότε το εσωτερικό for, ελέγχεται από το i του εξωτερικού for

Εμφάνισε 10, 9, 8,…, 1 αστεράκια (1)

- Κάντε το αντίστροφο, στην 1η γραμμή 10 αστεράκια, στην 2η σειρά 9 αστεράκια, κοκ, μέχρι την 10η σειρά με 1 αστεράκι
- Σκεπτόμαστε ότι όταν το εξωτερικό for θα έχει τιμή 1 (1η σειρά), το αστεράκια θα είναι 10. Όταν το εξωτερικό for θα έχει τιμή 2 (2η γραμμή) τα αστεράκια θα είναι 9 κλπ μέχρι τη 10η γραμμή, όπου το I του εξωτερικού θα είναι 10 και το εσωτερικό θα εκτυπώσει ένα αστεράκι

Εμφάνισε 10, 9, 8,…, 1 αστεράκια (2)

- Άρα και εδώ το εξωτερικό for θα ελέγχει το εσωτερικό αλλά με αντίστροφο τρόπο, αντί δηλαδή να είναι j <= i, πως μπορεί να είναι;
- Δείτε την επόμενη διαφάνεια

Εμφάνισε 10, 9, 8,…, 1 αστεράκια (3)


```java
package gr.aueb.cf.ch4;

/**
 * Εκτυπώνει 10 αστεράκια στη 1η σειρά,
 * 9 αστεράκια στη 2η σειρά, μέχρι 1
 * αστεράκι στην 10η σειρά.
 */
public class StarsDesc10 {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            for (int j = i; j <= 10; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

- Το εσωτερικό for τρέχει από i έως 10, οπότε την 1η φορά θα εκτελεστεί 10 φορές, την 2η 9 φορές, κλπ.

Εμφάνισε 10, 9, 8,…, 1 αστεράκια (4)


```java
package gr.aueb.cf.ch4;

/**
 * Εκτυπώνει 10 αστεράκια στη 1η σειρά,
 * 9 αστεράκια στη 2η σειρά, μέχρι 1
 * αστεράκι στν 10η σειρά.
 */
public class StarsDesc10 {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            for (int j = 10; j >= i; j--) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

- Εναλλακτικά μπορεί το j να ξεκινά από το 10 μέχρι το i και να μειώνεται j—
- Την 1η φορά θα εκτελεστεί 10 φορές και καθώς το i αυξάνει την 2η φορά θα εκτελεστεί από 10 έως 2, δηλαδή 9 φορές, την 3η φορά από 10 έως 3, δηλαδή 8 αστεράκια, κλπ.

Άσκηση

- Αναπτύξτε πέντε προγράμματα αντίστοιχα με τα προηγούμενα όπου ο χρήστης θα δίνει το πλήθος των stars, έστω n
  - n οριζόντια αστεράκια,
  - n κάθετα,
  - nxn,
  - από 1 έως n,
  - από n έως 1