7 CHARS

Terry Marris 16 April 2001

7.1 OBJECTIVES

By the end of this lesson the student should be able to

- use chars as fields, methods types and argument values
- appreciate that integer numbers represents chars
- understand the basis of the ASCII collating sequence

7.2 PRE-REQUSITES

The student should be familiar with classes, boolean and String fields, relational operators, parameters and arguments.

7.3 PREVIEW

We have already met the *int*, *double* and *boolean* primitive (or non-object) data types. We now look at the *char* primitive type.

Sometimes we might want to use a single character to represent a state e.g. *S* for small, *M* for medium, *L* for large and *X* for extra large. A variable of type *char* holds just a single character (letter, digit or punctuation symbol) at a time.

We look at the char data type, how a char value is stored in memory and the ASCII collating sequence.

7.4 CHAR

We have seen that variables of type *int* store a single integer. Variables of type *char* hold just a single character.

What is a character? Letters of the alphabet, punctuation symbols and digits are all examples of characters. *char* values are enclosed within single quotation marks like this.

```
char letterA = 'A';
char comma = ',';
char space =  ' ';
char digit5 = '5';
```

Note that *digit5* is not an integer; you cannot do sums with it.

7.5 ESCAPE SEQUENCES

What if you wanted to store a single quote itself? We use a backslash like this.

```
char singleQuote = '\'';
```

The `\'` looks like two characters but it represents just one - the single quote itself.

The `\'` is an example of an escape sequence - escape because the \ causes an escape from the normal meaning of whatever follows. The escape sequences are shown in the following table.

Escape sequence	Name
\b	backspace
\t	tab
\n	linefeed
\r	carriage return
\"	double quote
\'	single quote
\\	backslash

Figure 7.1 *The Escape Sequences*

The *t* (tab) is useful for tabbing output across a screen. For example

```
System.out.println("Number: " + number + "\tName: " + name);
```

might output

```
Number: 1473099          Name: Marris
```

The *t* in *"\tName"* causes *Name* to be displayed in the next tab position after *number*.

7.6 UNICODE AND ASCII COLLATING SEQUENCE

Computers store only number values. So a character has to be converted to a number before it can be stored in a computer's memory. This conversion, from *char* to number and back again is done automatically.

In the Unicode encoding scheme 'A is represented by 65, 'B' by 66, 'Z' by 90, 'a' by 97, 'z' by 122, the digit characters '0' by 48 and '9' by 57 and the space character by 32.

char	integer
space	32
-	45

char	integer
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

char	integer
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

char	integer
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Figure 7.2 *TheASCII Collating Sequence*

Some students might recognise these as the American Standard Code for Information Interchange (ASCII) values. Indeed, ASCII is a subset of Unicode. Unicode extends ASCII so that all characters in all languages might be represented. An ASCII table is shown in Appendix C.

Since characters are represented by number values, we can use the relational operators. For example

'A' < 'B'

because 'A' = 65 comes before 'B' = 66. This ordering is known as the ASCII collating sequence.

The ASCII collating sequence orders letters (and hence words) into something like dictionary order. For example:

Bat comes before *Cat* because 'B' comes before 'C' in the ASCII collating sequence.

But *bat* comes after *Cat* because 'b' comes after 'C'.

And again *to day* precedes *today* in the ASCII collating sequence because space comes before all letters.

7.7 THE AGENT CLASS

We use the *char* data type to represent an agent's gender (m for male, f for female) in the Agent class shown below.

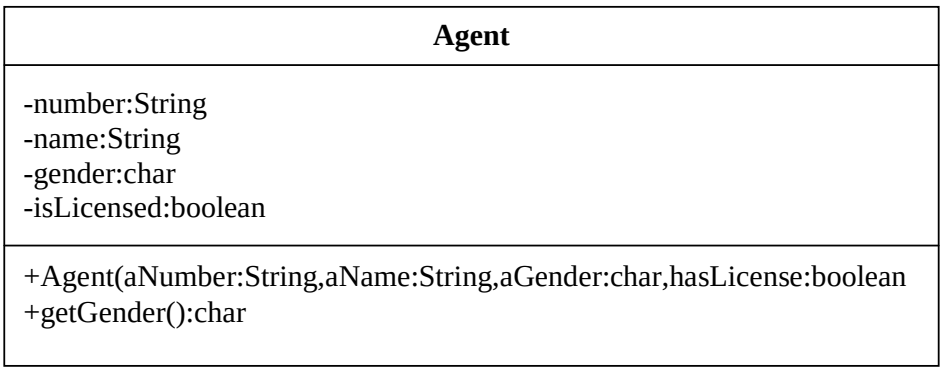


Figure 7.3 *The Agent class*

The implementation is very straightforward and is shown on the next page. The parts that refer to *char*s are shown emboldened.

```
/* Agent.java
   Terry Marris  16 April 2001
*/


public class Agent {
  private String number;
  private String name;
  private char gender;
  private boolean hasLicense;


  public Agent(String aNumber, String aName, char aGender,
               boolean isLicensed)
  {
    number = aNumber;
    name = aName;
    gender = aGender;
    hasLicense = isLicensed;
  }


  public char getGender()
  {
    return gender;
  }


  public static void main(String[] s)
  {
    Agent bond = new Agent("007", "Bond", 'm', true);
    System.out.println("Agent's gender is ... " +
                       bond.getGender());
  }
}
```

Program run:

```
Agent's gender is ... m
```

chars can be instance variables.

```
    private char gender;
```

They can appear as parameters.

```
  public Agent(String aNumber, String aName, char aGender,
               boolean isLicensed)
```

You can assign one *char* to another.

```
    gender = aGender;
```

You can return a *char* as a method value.

```
  public char getGender()
  {
    return gender;
  }
```

You can pass a *char* as an argument value.

```
    Agent bond = new Agent("007", "Bond", 'm', true);
```

You can print a *char* on the screen.

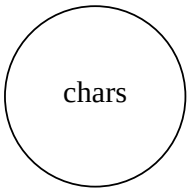
```
    System.out.println("Agent's gender is ... " +
                       bond.getGender());
```

For now we ignore the problem that 'm' ≠ 'M' ; we shall resolve the problem in Chapter 11 The Wrapper Classes.

7.8 FURTHER READING

HORSTMANN C & CORNELL G *Core Java 2 Volume 1 Fundamentals* pp 50

7.9 REVIEW



represent a single character

character

letter, digit, punctuation symbol

stored in memory as a number

space = 32, A = 65, Z = 90, a = 97, z = 122

ordered by the ASCII collating sequence

the relational operators, <, == and > can be used

7.10 EXERCISES

1 Explain, with the aid of an example, what is meant by

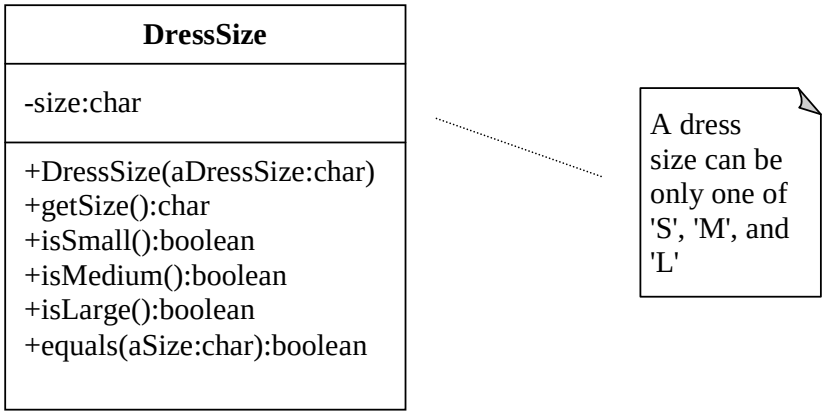
(a) *the char data type*
(b) the ASCII collating sequence
(c) Unicode

2 State the value of each of the following boolean expressions:

(a) `' ' < 'A'` **(b)** `'0' > 'Z'` **(c)** `'C' < 'c'`
(d) `'A' < 'Z'` **(e)** `'0' > '9'`

3 Implement and test the *DressSize* class specified below.

x *isMedium()* returns t*true* if the character stored in *size* is 'M', *false* otherwise.



The attribute *size* is to contain either 'S', 'M' or 'L'.

getSize() returns the character stored in the *size* attribute.

isSmall() returns *true* if the character stored in *size* is 'S', *false* otherwise.
isMedium() returns *true* if the character stored in *size* is 'M', *false* otherwise.
isLarge() returns *true* if the character stored in *size* is 'L', *false* otherwise.

equals(char) returns *true* if the given *aSize* is the same as the character stored in the *size* attribute.