14.5 Class `Character`

Java provides eight **type-wrapper classes**—`Boolean`, `Character`, `Double`, `Float`, `Byte`, `Short`, `Integer` and `Long`—that enable primitive-type values to be treated as objects. In this section, we present class `Character`—the type-wrapper class for primitive type `char`.

Most `Character` methods are `static` methods designed for convenience in processing individual `char` values. These methods take at least a character argument and perform either a test or a manipulation of the character. Class `Character` also contains a constructor that receives a `char` argument to initialize a `Character` object. Most of the methods of class `Character` are presented in the next three examples. For more information on class `Character` (and all the type-wrapper classes), see the `java.lang` package in the Java API documentation.

Figure 14.15 demonstrates `static` methods that test characters to determine whether they're a specific character type and the `static` methods that perform case conversions on characters. You can enter any character and apply the methods to the character.

```
1  // Fig. 14.15: StaticCharMethods.java
2  // Character static methods for testing character
```

```
3  import java.util.Scanner;
4
5  public class StaticCharMethods {
6     public static void main(String[] args) {
7        Scanner scanner = new Scanner(System.in); /
8        System.out.println("Enter a character and p
9           String input = scanner.next();
10       char c = input.charAt(0); // get input char
11
12          // display character info
13       System.out.printf("is defined: %b%n", Chara
14       System.out.printf("is digit: %b%n", Charact
15       System.out.printf("is first character in a
16          Character.isJavaIdentifierStart(c));
17       System.out.printf("is part of a Java identi
18          Character.isJavaIdentifierPart(c));
19       System.out.printf("is letter: %b%n", Charac
20          System.out.printf(
21          "is letter or digit: %b%n", Character.is
22          System.out.printf(
23          "is lower case: %b%n", Character.isLower
24          System.out.printf(
25          "is upper case: %b%n", Character.isUpper
26          System.out.printf(
27          "to upper case: %s%n", Character.toUpper
28          System.out.printf(
29          "to lower case: %s%n", Character.toLower
30       }
31    }
```

```
Enter a character and press Enter
A
is defined: true
is digit: false
is first character in a Java identifier: true
is part of a Java identifier: true
is letter: true
is letter or digit: true
```

```
is lower case: false
is upper case: true
to upper case: A
to lower case: a
```

```
Enter a character and press Enter
8
is defined: true
is digit: true
is first character in a Java identifier: false
is part of a Java identifier: true
is letter: false
is letter or digit: true
is lower case: false
is upper case: false
to upper case: 8
to lower case: 8
```

```
Enter a character and press Enter
$
is defined: true
is digit: false
is first character in a Java identifier: true
is part of a Java identifier: true
is letter: false
is letter or digit: false
is lower case: false
is upper case: false
to upper case: $
to lower case: $
```

Fig. 14.15

`Character static` methods for testing characters and converting case.

Line 13 uses `Character` method `isDefined` to determine whether character `c` is defined in the Unicode character set. If so, the method returns `true`; otherwise, it returns `false`. Line 14 uses `Character` method `isDigit` to determine whether character `c` is a defined Unicode digit. If so, the method returns `true`, and otherwise, `false`.

Line 16 uses `Character` method `isJavaIdentifierStart` to determine whether `c` is a character that can be the first character of an identifier in Java —that is, a letter, an underscore (_) or a dollar sign ($). If so, the method returns `true`, and otherwise, `false`. Line 18 uses `Character` method `isJavaIdentifierPart` to determine whether character `c` is a character that can be used in an identifier in Java—that is, a digit, a letter, an underscore (_) or a dollar sign ($). If so, the method returns `true`, and otherwise, `false`.

Line 19 uses `Character` method `isLetter` to determine whether character `c` is a letter. If so, the method returns `true`, and otherwise, `false`. Line 21 uses `Character` method `isLetterOrDigit` to determine whether character `c` is a letter or a digit. If so, the method returns `true`, and otherwise, `false`.

Line 23 uses `Character` method `isLowerCase` to determine whether `c` is a lowercase letter. If so, the method

returns `true`, and otherwise, `false`. Line 25 uses `isUpperCase` to determine whether `c` is an uppercase letter. If so, the method returns `true`, and otherwise, `false`. Line 27 uses `Character` method `toUpperCase` to convert the character `c` to its uppercase equivalent. The method returns the converted character if the character has an uppercase equivalent, and otherwise, the method returns its original argument. Line 29 uses `Character` method `toLowerCase` to convert the character `c` to its lowercase equivalent. The method returns the converted character if the character has a lowercase equivalent, and otherwise, the method returns its original argument.

Figure 14.16 demonstrates static `Character` methods `digit` and `forDigit`, which convert characters to digits and digits to characters, respectively, in different number systems. Common number systems include decimal (base 10), octal (base 8), hexadecimal (base 16) and binary (base 2). The base of a number is also known as its **radix**. For more information on conversions between number systems, see online Appendix J.

```
   1  // Fig. 14.16: StaticCharMethods2.java
 2  // Character class static conversion methods.
          3  import java.util.Scanner;
                        4
       5  public class StaticCharMethods2 {
  6     public static void main(String[] args) {
 7        Scanner scanner = new Scanner(System.in);
                        8
             9        // get radix
10        System.out.println("Please enter a radix:")
```

```
      11        int radix = scanner.nextInt();
                          12
             13        // get user choice
   14        System.out.printf("Please choose one:%n1 --
   15           "Convert digit to character", "Convert c
      16         int choice = scanner.nextInt();
                          17
             18        // process request
             19         switch (choice) {
   20           case 1: // convert digit to character
   21              System.out.println("Enter a digit:");
    22               int digit = scanner.nextInt();
   23              System.out.printf("Convert digit to c
   24                 Character.forDigit(digit, radix));
               25              break;
   26           case 2: // convert character to digit
   27              System.out.println("Enter a character
   28              char character = scanner.next().charA
   29              System.out.printf("Convert character
   30                 Character.digit(character, radix))
               31              break;
                    32         }
                      33     }
                        34  }
```

```
Please enter a radix:
16
Please choose one:
1 -- Convert digit to character
2 -- Convert character to digit
2
Enter a character:
A
Convert character to digit: 10
```

```
Please enter a radix:
```

```
                    16
          Please choose one:
1 -- Convert digit to character
2 -- Convert character to digit
                    1
            Enter a digit:
                   13
 Convert digit to character: d
```

Fig. 14.16

`Character` class `static` conversion methods.

Line 24 uses method `forDigit` to convert the integer `digit` into a character in the number system specified by the integer `radix` (the base of the number). For example, the decimal integer `13` in base 16 (the `radix`) has the character value `'d'`. Lowercase and uppercase letters represent the *same* value in number systems. Line 30 uses method `digit` to convert variable `character` into an integer in the number system specified by the integer `radix` (the base of the number). For example, the character `'A'` is the base 16 (the `radix`) representation of the base 10 value 10. The radix must be between 2 and 36, inclusive.

Figure 14.17 demonstrates the constructor and several instance methods of class `Character`—`charValue`, `toString` and `equals`. Lines 5–6 instantiate two `Character` objects by assigning the character constants `'A'` and `'a'`, respectively, to the `Character` variables. Java automatically

converts these `char` literals into `Character` objects—a process known as *autoboxing* that we discuss in more detail in Section 16.4. Line 9 uses `Character` method `charValue` to return the `char` value stored in `Character` object `c1`. Line 9 also gets a string representation of `Character` object `c2` using method `toString`. The condition in line 11 uses method `equals` to determine whether the object `c1` has the same contents as the object `c2` (i.e., the characters inside each object are equal).

```
1  // Fig. 14.17: OtherCharMethods.java
2  // Character class instance methods.
3  public class OtherCharMethods {
4     public static void main(String[] args) {
5        Character c1 = 'A';
6        Character c2 = 'a';
7
8        System.out.printf(
9           "c1 = %s%nc2 = %s%n%n",  c1.charValue(),
10
11       if (c1.equals(c2)
12          System.out.println("c1 and c2 are equal%
13       }
14       else {
15          System.out.println("c1 and c2 are not
16       }
17    }
18 }
```

```
c1 = A
c2 = a

c1 and c2 are not equal
```

Fig. 14.17

`Character` class instance methods.