APPENDIX B: JAVA KEYWORDS (RESERVED WORDS)

Keywords of a programming language are words that have a special meaning for the compiler of the language. Keywords may not be used as names in a program. They are thus "reserved words". All the Java keywords are listed and briefly explained below. We have not studied all the Java keywords in this book because some of them are needed only in very rare situations. The list below explains something about the keywords that we did not study.

abstract	The **abstract** keyword is a modifier that is used when abstract methods and abstract classes are declared. A class that contains an abstract method is an abstract class that serves as a base class for other classes. An abstract class cannot be instantiated.
boolean	This keyword identifies the boolean type that can get values **true** and **false**.
break	With a **break** statement it is possible to break out from a loop or a **switch-case** construct. I recommend that **break** statements should be used only in **switch-case** constructs. Using a **break** statement inside a loop is not logical. By using state variables it is possible to avoid **break** statements inside loops.
byte	8-bit integral type to store values in the range -128 ... 127.
case	This keyword is used in **switch-case** constructs.
catch	Keywords **catch**, **finally**, **throw**, **throws**, and **try** are needed when exceptions are handled in Java programs. Exceptions are "error objects" which Java programs can throw and catch in problem situations. **catch** begins the **catch** block of a **try-catch**(**-finally**) construct.
char	This keyword specifies the built-in (variable) type that has 16 bits, and can store the Unicode character code of a character.
class	This keyword begins class declarations.
const	**const** is a Java keyword that is not yet in use. The developers of Java have wanted to reserve this word for future use.
continue	A **continue** statement inside a loop causes a jump to the end of the internal statements of the loop. **continue** statements have not been studied in this book, and I recommend that they should not be used in programs because they are similar to **goto** statements. Instead of **continue** statements, more logical program constructs like **if** constructs should be favored.
default	This keyword is used inside **switch-case** constructs to identify the default case.
do	This keyword is used to build **do-while** loops that execute at least once.
double	The keyword that identifies the 8-byte double-precision floating-point type.
else	The keyword that is used in **if-else** constructs.
enum	This keyword can be used to specify enum types. An enum declaration specifies both an enum type and the constants that can be assigned as values to the type.
extends	With the **extends** keyword a class can be made to inherit another class.
false	A possible value for a variable of type **boolean**. The other possible **boolean** value is **true**.Officially, **false** and **true** are not keywords in Java, but they are **boolean** literals.
final	This keyword specifies that a local variable or a field is a constant whose value may not change. In addition, the **final** keyword can specify that a class is a "sealed" class from which it is not possible to derive new classes, or it can specify that a method may not be overridden in derived classes.

finally	Keyword **finally** begins the optional **finally** block of a **try-catch(-finally)** construct. The statements of a **finally** block are always executed regardless of whether or not an exception is caught.
float	The keyword used to identify the 4-byte single-precision floating-point type.
for	When the compiler sees this keyword, it knows that a **for** loop begins. In Java, the **for** keyword identifies both the traditional **for** loops and the "foreach" loops.
goto	This keyword has a bad reputation. By using this keyword it has traditionally been possible to write so-called **goto** statements with which it is possible to jump to a certain location in the program. It is one of the earliest findings in research related to computer programming that **goto** statements make programs illogical and difficult to understand. Therefore, **goto** statements should not be used in computer programs. **goto** statements can be avoided when state variables are used. Although **goto** is a keyword in Java, it does not work, i.e., it is not possible to write **goto** statements in Java.
if	This keyword begins an **if** construct.
implements	The **implements** keyword is used when a class implements one or more interfaces.
import	With this keyword a certain package can be imported to a program. Classes and other types in the package that is taken to use with an **import** statement can be referred to without mentioning the name of the package. An **import** statement can alternatively be written so that it imports only a single class of a package.
instanceof	A relational operator with which it is possible to check if something is of certain type. The **instanceof** operator returns **true** when the type of its left operand is the type specified by the right operand, or when the type of the left operand is derived from or implements the type specified by the right operand.
int	This keyword specifies that a variable, a field, a method, etc. is of the basic integer type. **int** is a 32-bit (4-byte) type.
interface	A keyword that begins an interface declaration. When a class implements an interface, it provides implementations for the methods that are specified in the declaration of the interface.
long	Type **long** is an integral type that uses 8 bytes (64 bits) to store integer values.
native	The **native** keyword specifies that a method is a native method, which is implemented so that it is compiled to native machine instructions of the computer. Usually native methods are written with a programming language other than Java, and compiled with the compiler of the used language. Native methods can be called from a Java program when method declarators are introduced with the **native** keyword.
new	An operator that is used to create objects. The **new** operator allocates memory space from the heap memory and invokes a constructor.
null	**null** means that no object is being referenced. This literal can be assigned to object references. **null** is the default value when object references are fields of a class or array elements.
package	A keyword that can be used to specify that certain classes, enums, and interfaces belong to a certain named package.
private	This keyword is an access modifier which specifies that a class member is accessible only in the class in which it is declared.
protected	This access modifier specifies that a class member is accessible for methods in classes that belong to the same package as well as for methods in derived classes in other packages.

public	The keyword to specify public members in a class declaration. Public members can be accessed by methods of all classes in all packages. If a type (e.g. class, enum, or interface) is marked with the **public** keyword, the type is accessible to all types outside the package of the type. A package can contain a set of types, and those types that are not marked as **public** are accessible only within the package.
return	A statement that causes a return to the calling method. Usually **return** statements supply a value to the calling method. Because a **return** statement is a kind of **goto** statement that causes a jump to the end of a method, **return** statements should be used sparingly and only at the end of methods.
short	An integral type that uses 2 bytes (16 bits) to store values in the range -32,768 ... 32,767.
static	A keyword to specify that a method or a field is static. A static method can be called without creating an object of a class. A static field is shared by all objects of a class.
super	With the **super** keyword it is possible to call the constructors and polymorphic methods of the superclass.
switch	**switch-case** program constructs begin with the **switch** keyword.
synchronized	The **synchronized** keyword can be used in programs which run several threads simultaneously, and in which a method is called from several threads. When a method is declared with the **synchronized** keyword, only one thread at a time can use it. Other threads automatically wait until a thread has executed the code of a **synchronized** method. If it is not possible to make an entire method a **synchronized** method, a set of statements of a method can be put inside a **synchronized** block of statements.
this	When the **this** keyword is used inside a method, it references the object for which the method was called. With the **this** keyword, a constructor can call another constructor of the same class.
throw	A **throw** statement throws an exception object in an error situation. A **throw** statement results in a jump to the nearest **catch** block where the exception can be handled.
throws	With the **throws** keyword it is possible to write a so-called exception specification, which specifies exceptions that may be thrown by a method.
transient	A field of a class can be marked with the **transient** keyword to indicate that its value does not need to be stored when an object of the class is serialized. Serialization of objects means that the data stored in an object is converted to such a form that it is easy to store it, for example, to a file. Object serialization is not covered by this book.
true	A value that can be given for a variable of type **boolean**. Officially, **true** is not a keyword in Java, but it is a **boolean** literal.
try	Keyword **try** begins a **try-catch**(**-finally**) construct.
void	When written in a method declaration, this keyword says that the method does not return a value with a **return** statement.
volatile	A field of a class can be declared **volatile** in order to increase reliability of applications that consist of several threads.
while	This keyword is needed when **while** loops and **do-while** loops are written.