

# **Data Storage and Its Implications**

In this exercise you will:

1. Investigate the storage systems used for three primitive data types, `int`, `byte` and `character`.
2. Learn how these storage systems are reflected in the execution of Java programs.

## Data of Type Integer

Recall that data items of type integer are normally stored in memory by using two's complement notation and that this imposes a limit on the size of the values that can be represented. For example, with a two's complement system using only four bits for storage, the largest value that can be represented is seven. Of course, if this was the largest integer your machine could represent, it would be of little use. Therefore, computer systems use more than four bits for integer storage and can store rather large values. But, limitations still exist.

In the Java system, the maximum and minimum values of the numeric types are represented by the constants `MAX_VALUE` and `MIN_VALUE`. The smallest and largest `int` values are defined in the `Integer` class. These class constants can be accessed by using the name of the class in which the constant is defined, followed by the dot operator (`.`) and the name of the constant. Therefore, to access the largest and smallest `int` values, use `Integer.MAX_VALUE` and `Integer.MIN_VALUE`.

### Experiment 3.1

In this experiment you will investigate the storage of integer values in Java.

**Step 1.** Compile and execute the program `J03E01.java`, and record the values of the smallest and largest `int` values.

```
class J03E01
{
    public static void main(String[] args)
    {
        System.out.println(
            "In Java the smallest int value is " + Integer.MIN_VALUE);
        System.out.println(
            "In Java the largest int value is " + Integer.MAX_VALUE);
    }
}
```

---



---



---



---



---



---



---



---

**Step 2.** Besides `int`, Java has three other integer types, `long`, `short` and `byte`. In this step, let's investigate the maximum and minimum values of the primitive data type `byte`. Information about and methods involving type `byte` are stored in a class called `Byte`. Both the `Integer` and `Byte` classes are in the API package `java.lang`, which is automatically imported into every Java program. Modify the program to print the smallest and largest `byte` values. Record the results.

---

---

---

---

---

---

---

**Step 3.** Since integers are represented in two's complement notation, `MAX_VALUE` should be one less than some power of two,  $2^n - 1$ , and `MIN_VALUE` should be a power of 2,  $2^n$ , for some integer  $n$ . Explain why.

---

---

---

---

---

---

---

**Step 4.** For what value of  $n$  is  $2^n - 1$  equal to `Byte.MAX_VALUE` and  $-2^n$  equal to `Byte.MIN_VALUE`?

---

---

---

---

---

**Step 5.** Use the information collected in Steps 2 through 4 to determine the number of bits used to represent data of type `byte`. How many bytes are used? Explain your answers.

---

---

---

---

---

---

---

## Data of Type Character

In Java, data items of type character are stored as bit patterns according to the Unicode encoding scheme. ASCII is an older eight-bit code whereas Unicode is a 16-bit code.

*Your textbook contains a partial listing of the Unicode character set in the appendix.*

In a sense, this appendix is also a partial listing of ASCII because the ASCII codes are a subset of Unicode. Each ASCII character becomes Unicode when 8 zeros are added to the front.

Thus, since the character *a* is represented as

01100001 in ASCII, it is represented as 0000000001100001 in Unicode.

The following program (J03E02) illustrates how we can identify the bit pattern used to represent a character. First note that if the variable *ch* (of type *char*) is assigned the value '*a*', then the statement

```
System.out.print(ch);
```

prints the character *a*. We can, however, request that the bit pattern assigned to *ch* be interpreted as though the variable was of type *int* via the statement

```
System.out.println((int)ch);
```

Here the expression

```
(int)ch
```

is an example of casting. In general, a cast takes the form

*(newDataType) variable*

where *newDataType* is the data type to be applied to the bit pattern assigned to *variable*. Therefore, if the variable *ch* (of type *char*) is assigned the value 'a', then the statement

```
System.out.println((int)ch);
```

prints the integer 97, which is the value obtained when the bit pattern 0000000001100001 (the Unicode representation of the symbol *a*) is interpreted as a two's complement representation.

### Experiment 3.2

**Step 1.** Compile and execute J03E02.java. Record the results.

```
class J03E02
{
    public static void main(String[] args)
    {
        char ch = 'a';
        System.out.println(ch + " " + (int)ch);
        ch = '2';
        System.out.println(ch + " " + (int)ch);
        ch = 'Z';
        System.out.println(ch + " " + (int)ch);
    }
}
```

---



---



---



---



---

**Step 2.** Modify the program in Step 1 to find the Unicode representations for the characters ?, 0 and space. Summarize your work and findings below.

---



---



---



---



---



---



---



---

**Casting Experiment 3.3**

**Step 1.** Compile the program J03E03.java. Record and explain the results.

```
class J03E03
{
    public static void main(String[] args)
    {
        int x;
        double d;

        x = 5;
        d = x;
        System.out.println("int " + x + "    double " + d);

        d = 2.95;
        x = d;
        System.out.println("int " + x + "    double " + d);
    }
}
```

---

---

---

---

---

---

---

---

---

---

**Step 2.** Correct the program in Step 1 by replacing the statement

```
x = d;
```

with

```
x = (int)d;
```

Record the results.

---

---

---

---

---