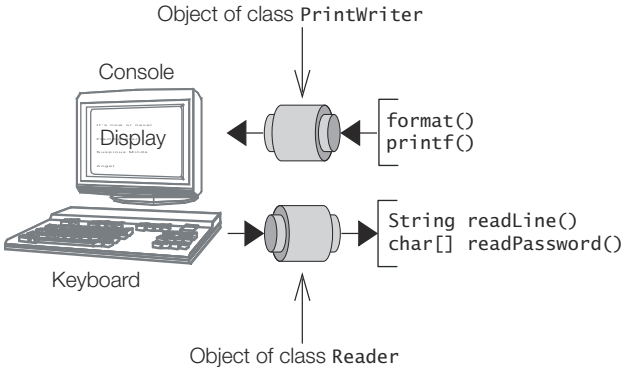
20.4 The `Console` Class

A *console* is a unique *character-based* device associated with a JVM. Whether a JVM has a console depends on the platform, and also on the manner in which the JVM is invoked. When the JVM is started from a command line, and the standard input and output streams have not been redirected, the console will normally correspond to the keyboard and the display (Figure 20.8). In any case, the console will be represented by an instance of the class `java.io.Console`. This `Console` instance is a *singleton*, and can only be obtained by calling the static method `console()` of the `System` class. If there is no console associated with the JVM, the `null` value is returned by this method.

```
// Obtaining the console:
Console console = System.console();
if (console == null) {
  System.err.println("No console available.");
  return;
}
// Continue ...
```

Figure 20.8 *Keyboard and Display as Console*



For creating dialogue for console-based applications, the `Console` class provides the following functionality:

- Prompt and read a line of character-based response.

  ```
  String username = console.readLine("Enter the username (%d chars): ", 4);
  ```

  The `readLine()` method first prints the formatted prompt on the console, and then returns the characters typed at the console when the line is terminated by the ENTER key.

- Prompt and read passwords without echoing the characters on the console.

  ```
  char[] password;
  do {
      password = console.readPassword("Enter password (min. %d chars): ", 6);
  } while (password.length < 6);
  ```

The readPassword() method first prints the formatted prompt, and returns the password characters typed by the user in an array of char when the line is terminated by the ENTER key. The password characters are not echoed on the display.

Since a password is sensitive data, one recommended practice is to have it stored in memory for only as long as it is necessary and to zero-fill the char array as soon as possible in order to overwrite the password characters.

- Print formatted values to the console.

  Similar to the PrintWriter and the PrintStream classes, the Console class also provides the format() and the printf() methods for printing formatted values, but its methods do *not* allow a locale to be specified.

Note that the console only returns character-based input. For reading other types of values from the standard input stream, the java.util.Scanner class can be considered.

The Console class provides methods for *formatted prompting* and *reading* from the console, and obtaining the reader associated with it.

```
String readLine()
String readLine(String format, Object... args)
```

The first method reads a single line of text from the console. The second method prints a formatted prompt first, then reads a single line of text from the console. The prompt is constructed by formatting the specified args according to the specified format.

```
char[] readPassword()
char[] readPassword(String format, Object... args)
```

The first method reads a password or a password phrase from the console with echoing disabled. The second method does the same, but first prints a formatted prompt.

```
Reader reader()
```

This retrieves the unique Reader object associated with this console.

The Console class provides the following methods for *writing* formatted strings to the console, and obtaining the writer associated with it:

```
Console format(String format, Object... args)
Console printf(String format, Object... args)
```

Write a formatted string to this console's output stream using the specified format string and arguments, according to the default locale. See the PrintWriter class with analogous methods (p. 1245).

```
PrintWriter writer()
```

Retrieves the unique PrintWriter object associated with this console.

```
void flush()
```

Flushes the console and forces any buffered output to be written immediately.

Example 20.5 illustrates using the `Console` class to change a password. The example illustrates the capability of the `Console` class, and in no way should be construed to provide the ultimate secure implementation to change a password.

The console is obtained at (1). The code at (2) implements the procedure for changing the password. The user is asked to submit the new password, and then asked to confirm it. Note that the password characters are not echoed. The respective `char` arrays returned with this input are compared for equality by the static method `equals()` in the `java.util.Arrays` class, which compares two arrays.

Example 20.5 *Changing Passwords*

```
import java.io.Console;
import java.io.IOException;
import java.util.Arrays;

/** Class to change the password of a user */
public class ChangePassword {
  public static void main (String[] args) throws IOException  {

    // Obtain the console:                                          (1)
    Console console = System.console();
    if (console == null) {
      System.err.println("No console available.");
      return;
    }

    // Changing the password:                                       (2)
    boolean noMatch = false;
    do {
      // Read the new password and its confirmation:
      char[] newPasswordSelected
          = console.readPassword("Enter your new password: ");
      char[] newPasswordConfirmed
          = console.readPassword("Confirm your new password: ");

      // Compare the supplied passwords:
      noMatch = newPasswordSelected.length == 0  ||
                newPasswordConfirmed.length == 0 ||
                !Arrays.equals(newPasswordSelected, newPasswordConfirmed);
      if (noMatch) {
        console.format("Passwords don't match. Please try again.%n");
      } else {
        // Necessary code to change the password.
        console.format("Password changed.");
      }
    } while (noMatch);
  }
}
```

Running the program:

```
>java ChangePassword
Enter your new password:
```

```
Confirm your new password:
Password changed.
```

Review Questions

20.1 Given the following code, under which circumstance will the method return `false`?

```
public static boolean test(InputStream is) throws IOException {
  int value = is.read();
  return value >= 0;
}
```

Select the one correct answer.

(a) A character of more than 8 bits was read from the input stream.
(b) An I/O error occurred.
(c) This method will never return `false`.
(d) The end of the stream was reached in the input stream.

20.2 How can we programmatically check whether a call to a `print()` method of the `PrintWriter` class was successful or not?
Select the one correct answer.

(a) Check if the return value from the call is `-1`.
(b) Check if the return value from the call is `null`.
(c) Catch the `IOException` that is thrown when an I/O error occurs.
(d) Call the `checkError()` method of the `PrintWriter` class immediately after the `print()` method call returns to see if an `IOException` was thrown.

20.3 Given the following program:

```
import java.io.*;
public class MoreEndings {
  public static void main(String[] args) {
    try (FileInputStream fis = new FileInputStream("seq.txt");
        InputStreamReader isr = new InputStreamReader(fis);) {
      int i = isr.read();
      while (i != -1) {
        System.out.print((char)i + "|");
        i = isr.read();
      }
    } catch (FileNotFoundException fnf) {
      System.out.println("File not found");
    } catch (EOFException eofe) {
      System.out.println("End of stream");
    } catch (IOException ioe) {
      System.out.println("Input error");
    }
  }
}
```

Assume that the file `seq.txt` exists in the current directory, has the required access permissions, and contains the string `"Hello"`.

Which statement about the program is true?
Select the one correct answer.

(a) The program will fail to compile because a certain unchecked exception is not caught.
(b) The program will compile and print `H|e|l|l|o|Input error`.
(c) The program will compile and print `H|e|l|l|o|End of stream`.
(d) The program will compile, print `H|e|l|l|o|`, and then terminate normally.
(e) The program will compile, print `H|e|l|l|o|`, and then block in order to read from the file.
(f) The program will compile, print `H|e|l|l|o|`, and terminate because of an uncaught exception.

20.4 Given the following program:

```
import java.io.*;

public class NoEndings {
  public static void main(String[] args) {
    try (FileReader fr = new FileReader("greetings.txt");
         BufferedReader br = new BufferedReader(fr)) {
      System.out.print(br.readLine() + "|");
      System.out.print(br.readLine() + "|");
      System.out.print(br.readLine() + "|");
    } catch (EOFException eofe) {
      System.out.println("End of stream");
    } catch (IOException ioe) {
      System.out.println("Input error");
    }
  }
}
```

Assume that the file `greeting.txt` exists in the current directory, has the required access permissions, and contains the following two text lines:

```
Hello
Howdy
```

Which statement is true about the program?
Select the one correct answer.

(a) The program will fail to compile because the `FileNotFoundException` is not caught.
(b) The program will compile, print `Hello|Howdy|null|`, and then terminate normally.
(c) The program will compile and print `Hello|Howdy|Input error`.
(d) The program will compile and print `Hello|Howdy|End of stream`.
(e) The program will compile, print `Hello|Howdy|`, and then block in order to read from the file.
(f) The program will compile, print `Hello|Howdy|`, and terminate because of an uncaught exception.