Objectives

- In this session, you will learn to:
  - Write a program that uses command-line arguments and system properties
  - Write a program that reads from standard input
  - Describe the C-type formatted input and output
  - Write a program that can create, read, and write files
  - Describe the basic hierarchy of collections
  - Write a program that uses sets and lists
  - Write a program to iterate over a collection
  - Write a program that uses generic collections

Command-Line Arguments

- Command-line arguments are the parameters passed to a Java application at run time.
- Each command-line argument is placed in the `args` array that is passed to the static main method. For example:

```
public static void main(String[] args)
```

System Properties

- System properties are a feature that replaces the concept of environment variables (which are platform-specific).
- System properties include information about the current user, the current version of the Java runtime, and the character used to separate components of a file path name.
- The `System.getProperties()` method returns a Properties object.
  - The `System.getProperty(String)` method returns a String representing the value of the named property.
  - The `System.getProperty(String, String)` method enables you to supply a default string value (second parameter), which is returned if the named property does not exist.

Console I/O

- Applications interact with the user using console I/O.
- Java 2 SDK supports console I/O with three public variables in the `java.lang.System` class:
  - The variable `System.out` enables you to write to standard output. It is an object of type PrintStream.
  - The variable `System.in` enables you to read from standard input. It is an object of type InputStream.
  - The variable `System.err` enables you to write to standard error. It is an object of type PrintStream.

Writing to Standard Output

- The `println()` method print the argument and a newline character (\n).
- The `print()` method print the argument without a newline character.
- The `print()` and `println()` methods are overloaded for most primitive types (`boolean, char, int, long, float, and double`) and for `char[], Object`, and `String`.
- The `print(Object)` and `println(Object)` methods call the `toString()` method on the argument.

Reading from Standard Input

- The application program can use the following methods of the `java.io` package to read from the standard input:
  - Read characters from the keyboard and convert the raw bytes into Unicode characters:

    ```
    InputStreamReader ir=new
    InputStreamReader(system.in);
    ```

  - Create a buffered reader to read each line from the keyboard:

    ```
    BufferedReader in = new BufferedReader(ir);
    ```

  - The BufferedReader(in) provides a `readLine()` method to read from standard input one line at a time:

    ```
    s=in.readLine();
    ```

Files and File I/O

- The `java.io` package enables you to do the following:
  - Create File objects
  - Manipulate File objects
  - Read and write to file streams

Files and File I/O (Contd.)

- Creating a new `File` Object:

```
File myFile;
```

- The `File` class provides several utilities:

```
myFile = new File("myfile.txt");
myFile = new File("MyDocs", "myfile.txt");
```

- Directories are treated just like files in Java; the `File` class supports methods for retrieving an array of files in the directory, as follows:

```
File myDir = new File("MyDocs");
myFile = new File(myDir, "myfile.txt");
```

Files and File I/O (Contd.)

- For file input:
  - Use the `FileReader` class to read characters.
  - Use the `BufferedReader` class to use the `readLine()` method.
- For file output:
  - Use the `FileWriter` class to write characters.
  - Use the `PrintWriter` class to use the `print()` and `println()` methods.

Files and File I/O (Contd.)

- The application program can use the following methods of the `java.io` package to read input lines from the keyboard and write each line to a file:
  - Create file

    ```
    File file = new File(args[0]);
    ```
  - Create a buffered reader to read each line from the keyboard

    ```
    InputStreamReader isr=new
    InputStreamReader(System.in);
    BufferedReader in = new BufferedReader(isr);
    ```
  - Create a print writer on this file

    ```
    PrintWriter out = new PrintWriter(new
    FileWriter(file));
    ```

Files and File I/O (Contd.)

- Read each line from the input stream and print to a file one line at a time:

```
s = in.readLine();
out.println(s);
```

- The application program can use the following methods of the `java.io` package to read from a text file and display each line on the standard output.
  - Create file:

```
File file = new File(args[0]);
```

  - Create a buffered reader to read each line from the keyboard:

```
BufferedReader in = new BufferedReader(new
FileReader(file);
```

Files and File I/O (Contd.)

- Read each line from the file and displays it on the standard output:

```
s = in.readLine();
System.out.println(s);
```

Java Programming Language

Demonstration

Lets see how to read data from a file and display the output on the standard output device. This demo also shows how to run a program with user provided command line arguments.