**Administrative**

**Today’s session**

System class

Variable-length argument list

Runnable Java archive (JAR) file

*JFreeChart*

**Session Topics**

**System class**

● The **System class** has a range of computer-based functions.

● System defines the following three input/output streams:

| Field | Stream |
| --- | --- |
| System.in | Input stream, usually coming from the keyboard. |
| System.out | Output stream going to the screen. |
| System.err | Output stream also going to the screen. In Eclipse, the text appears in red. |

● See **System class** sample application on Blackboard.

● Here are some **static** System methods. See the complete list at [docs.oracle.com/javase/8/docs/api/java/lang/System.html](https://docs.oracle.com/javase/8/docs/api/java/lang/System.html).

| Method | Purpose |
| --- | --- |
| arrayCopy(<src-array>, <src-array-index>, <tgt-array>, <tgt-array-index>, <length>) | Copy <length> array elements from <src-array> to <tgt-array>. |
| currentTimeMillis() | Returns the current time in milliseconds. |
| exit(<int-return-code>) | End the application and return <int-return-code> to the operating system. |
| nanoTime() | Returns the current value of the running Java Virtual Machine's time in nanoseconds. |
| getProperty("<property>") | Return the value of the specified <property>. |
| getProperties() | Return the value of all properties. |

**Variable-length argument list**

● A method may be defined to accept a variable number of arguments of the same data type.

● A variable-length parameter has syntax:

… <method-name>(<parameter-list>, <variable-length parameter>)

{

<block>

}

Where:

**<parameter-list>** is a list of zero or more inputs needed by the method.

**<variable-length parameter>** must be the last parameter in the list.

● Since a variable-length parameter must be the last parameter, there may only be one such parameter.

● When the method is called, the list of arguments is stored in an array and made available to the method.

● The size of the array matches the number of arguments in the list.

● See **Variable length argument lists** sample application on Blackboard.

**Runnable Java archive (JAR) file**

● A JAR file is used to group and distribute Java files.

● A JAR file has a format similar to a ZIP file.

● A runnable JAR file is used to distribute a Java application.

● Eclipse enables the creation of JAR files.

● See document **Using Eclipse** on Blackboard.

● To run a runnable JAR file for a:

✓ Java GUI application, navigate to the file and double-click it.

✓ Java console application, open a command window, and …

→ Navigate to the folder containing the file, and enter command **java -jar <jar-file>.jar**.

OR

→ Enter command **java -jar** and drag the JAR file to the command window.

● Information may be passed into the runnable JAR file via the **args** array:

**java -jar <jar-file>.jar <arg-1> <arg-2> … <arg-n>**

● If a Java application reads input files and/or writes output files, the default folder used is the “current” operating system folder. This may be overridden by placing the JAR file and required (relative) folders and input files within a ZIP file and distributing that instead of the JAR file.

**JFreeChart**

● JFreeChart is a chart library for representing data graphically.

● Many chart types may be created with JFreeChart including:

|  |  |
| --- | --- |
| Chart type | Example |
| Bar chart |  |
| Bubble chart |  |
| Line chart |  |
| Pie chart |  |
| XY/scatter chart |  |

● To install JFreeChart on Eclipse, see document **Using Eclipse** on Blackboard.

● JFreeChart accepts data in the form of a JFreeChart dataset.

● The dataset is then charted and placed in a panel.

● See **Chart drawing** sample application on Blackboard.

**Midterm exam review**

● The Midterm exam review document is available on Blackboard.