The hsa_ufa Console: Keyboard Input

Getting Keyboard Input: The Strategy

Here is a very basic strategy for getting keyboard input from the user:

1. Halt the program.
2. Wait for the user to type some characters.
3. Turn those characters into a data type Java can understand.
4. Store the input in a variable.

For example, if you want to have the user enter something you can interpret as an `int`, you have to stop, wait for the user to type characters, and then turn those characters into a number in the `int` format.

Fortunately, the hsa_ufa Console contains special methods that take care of all that.

An Example: readInt()

If you want the user to enter an integer, here is what you should do:

1. Create a variable to store user input.
2. Prompt the user to enter something (a prompt is a message, like "Please enter your age.")
3. Call `c.readInt()` and store the result in the variable you created.

```
int age;
c.println("Please enter your age.");
age = c.readInt();
c.println("You entered: " + age);
```

A New Kind of Error: Run -Time Errors

There are three types of errors possible in any program. You are probably already familiar with two of them:

Syntax Errors
: These are **errors that prevent compilation** (for example, a missing semicolon or extra bracket).

Logic Errors
: The program compiles and runs without crashing, but it **doesn't do what you expected**.

The third kind of error can be caused by user input:

Run-Time Errors
: These are **errors that crash the program while it is running**. User input can cause run-time errors in a program (for example, you want an integer, but the user enters letters). There are other ways to cause them as well (for example, division by zero). In Ready to Program, run-time errors halt the program and cause an error message to pop up.

Some Basic Input Methods

There has to be a different method for every variable type. Here are three for data types you already know about (more to come).

Remember that all these methods require a "`c.`" (or the console name, if it's not "c") in front of them.

readInt ()
: Waits for the user to type and hit enter, then returns the number the user typed as an **`int`**. Run-time error if the user types something other than an `int` (like "6.7" or "4g5")

readDouble ()
: Waits for the user to type and hit enter, then returns the number the user typed as a **`double`**. Run-time error if something other than a `double` (like "4g5" or "7.8.9")

readLine()
: Waits for the user to type and hit enter, then returns the entire line as a **`String`**.

getChar()
: Waits for the user to press a button and then returns the key pressed as a **`char`**.

Some More Input Methods

The above four methods are all you are required to use for this course, but for completeness, herre are the rest of them…

readByte ()
: Waits for the user to type and hit enter, then returns the number the user typed as a **`byte`**. Run-time error if the user types something other than a **`byte`** (like "6.7" or "3290")

readShort ()
: Waits for the user to type and hit enter, then returns the number the user typed as a **`short`**. Run-time error the user types something other than a `short`.

readLong ()
: Waits for the user to type and hit enter, then returns the number the user typed as a **`long`**. Run-time error if something the user types something other than `long`.

readFloat ()
: Waits for the user to type and hit enter, then returns the number the user typed as a **`float`**. Run-time error if the user types something other than a `float.`

readBoolean ()
: Waits for the user to type and hit enter, then returns the number the user typed as a **`long`**. Run-time error if the user types something other than a `long`.

readToken()
: Returns the next token typed as a **`String`**. A token is a single word. This is useful for processing one word of input at a time, but **don't use this unless you really know what you're doing (ask the teacher first).**