Essential Computing 1

# Command-line arguments

# Arguments passed to the program when it is started

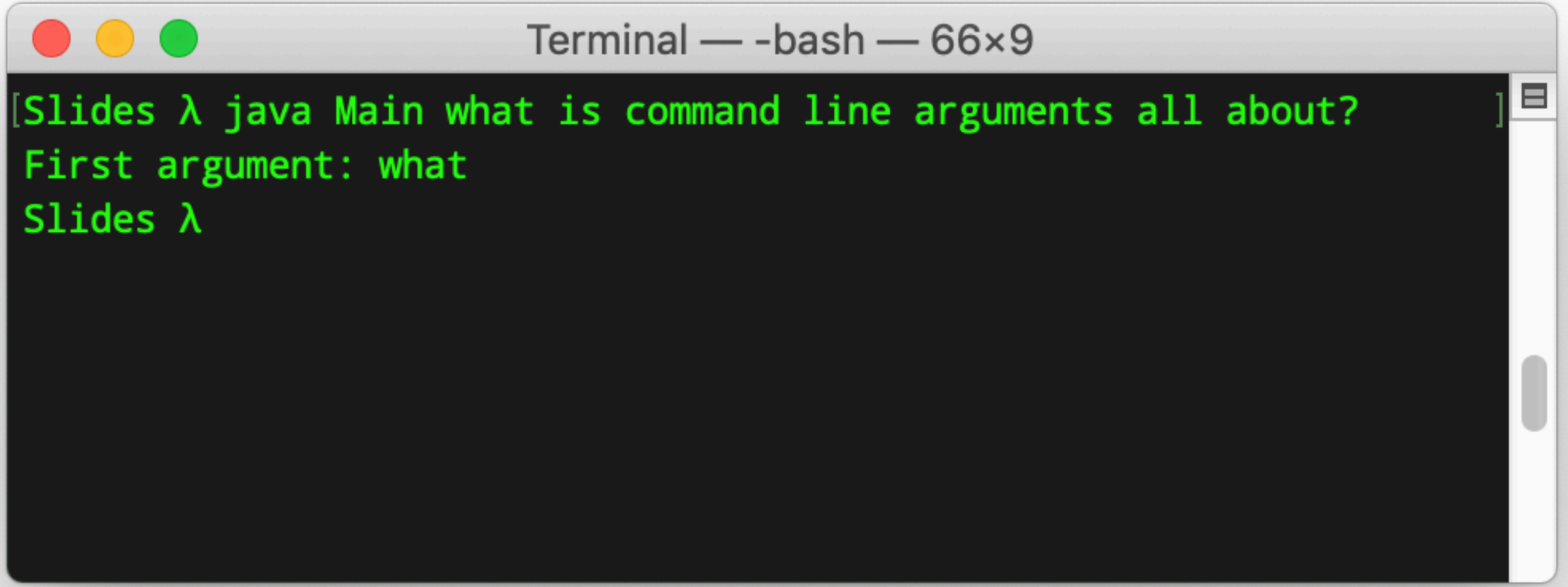
```
public class Main {
    public static void main( String[] args ){
    }
}
```

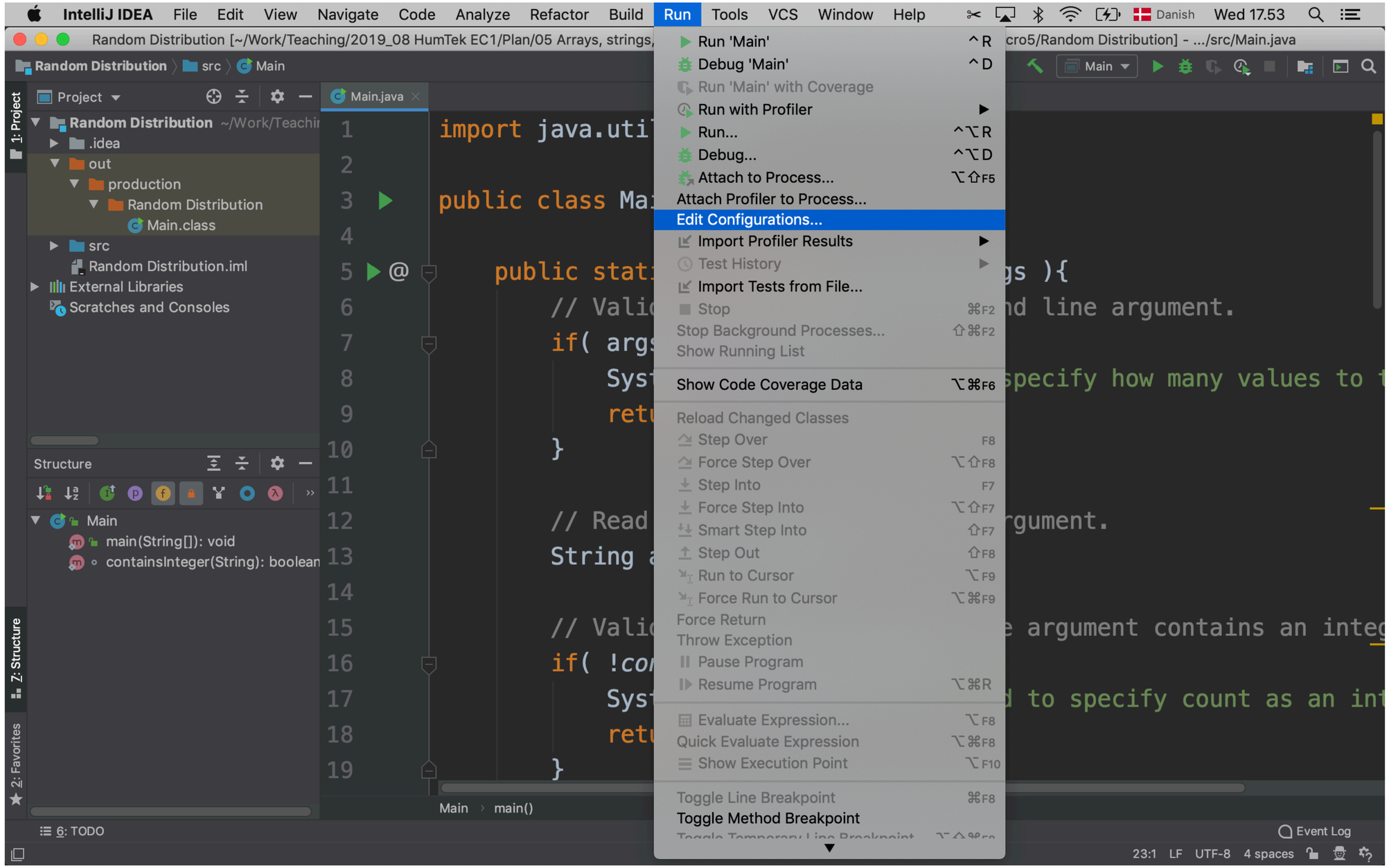
Let's try reading them (after validating that we have any)

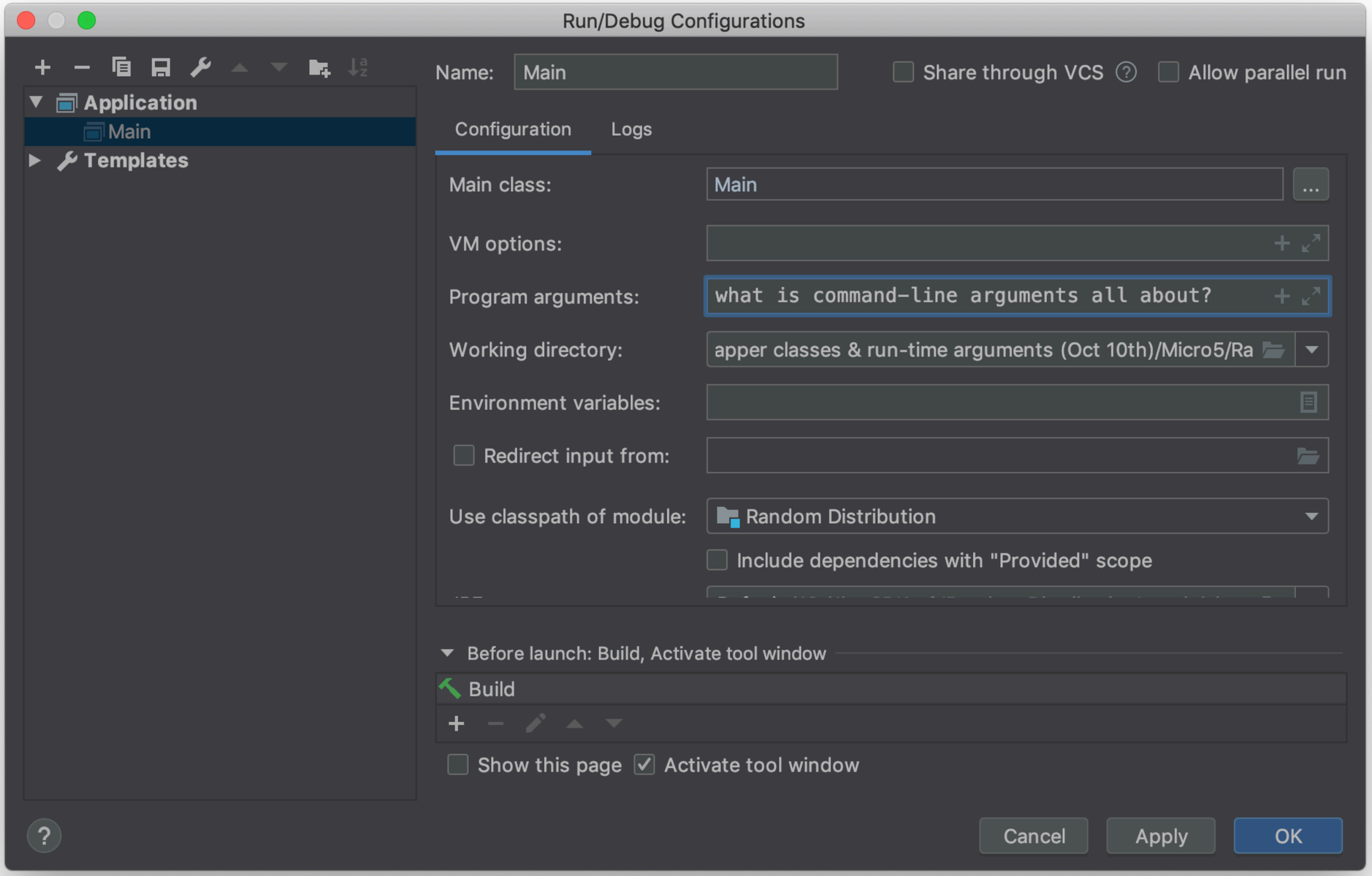
```
public class Main {

    public static void main( String[] args ){
        if( args.length > 0 ){
            System.out.println( "First argument: " + args[0] );
        }
    }

}
```


Terminal — -bash — 66×9
[Slides λ java Main what is command line arguments all about? ]
First argument: what
Slides λ


IntelliJ IDEA File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Random Distribution
Project
Main.java
Structure
Run 'Main' ^R
Debug 'Main' ^D
Run 'Main' with Coverage
Run with Profiler
Run... ^⌥R
Debug... ^⌥D
Attach to Process... ⌥⇧F5
Attach Profiler to Process...
Edit Configurations...
Import Profiler Results
Test History
Import Tests from File...
Stop ⌘F2
Stop Background Processes... ⇧⌘F2
Show Running List
Show Code Coverage Data ⌥⌘F6
Reload Changed Classes
Step Over F8
Force Step Over ⌥⇧F8
Step Into F7
Force Step Into ⌥⇧F7
Smart Step Into ⇧F7
Step Out ⇧F8
Run to Cursor ⌥F9
Force Run to Cursor ⌥⌘F9
Force Return
Throw Exception
Pause Program
Resume Program ⌥⌘R
Evaluate Expression... ⌥F8
Quick Evaluate Expression ⌥⌘F8
Show Execution Point ⌥F10
Toggle Line Breakpoint ⌘F8
Toggle Method Breakpoint
6: TODO
Event Log
23:1 LF UTF-8 4 spaces


Run/Debug Configurations
Application
Main
Templates
Name:
Main
Share through VCS
Allow parallel run
Configuration
Logs
Main class:
Main
VM options:
Program arguments:
what is command-line arguments all about?
Working directory:
apper classes & run-time arguments (Oct 10th)/Micro5/Ra
Environment variables:
Redirect input from:
Use classpath of module:
Random Distribution
Include dependencies with "Provided" scope
Before launch: Build, Activate tool window
Build
Show this page
Activate tool window
Cancel
Apply
OK

# How to **parse** from String to other datatypes?

```
public class Main {

    public static void main( String[] args ){
        if( args.length > 0 ){
            int number = args[0];
        }
    }

}
```

**Will not compile!**

Use **parsing methods** in the **primitive data type wrappers**.

```
public class Main {

    public static void main( String[] args ){
        if( args.length > 0 ){
            int number = Integer.parseInt( args[0] );
        }
    }

}
```

# Will throw a runtime exception if args[0] contains non-digits

```
public class Main {

    public static void main( String[] args ){
        if( args.length > 0 ){
            int number = Integer.parseInt( args[0] );
        }
    }

}
```

**java.lang.NumberFormatException: For input string: "arg0"**

The think Java book does not mention **try-catch**.
But that is what you need to deal with exceptions.

```
public class Main {
    public static void main( String[] args ){
        if( args.length > 0 ){
            try {
                int number = Integer.parseInt( args[0] );
            } catch ( Exception e ){
                // Deal with exception here
                // without crashing program.
            }
        }
    }
}
```

Alternatively, use a method to validate before reading.

```
/**
 * Returns true if string contains an integer.
 * @param text The text to evaluate.
 * @return Boolean flag indicating if text contains an integer.
 */
static boolean containsInteger( String text ){
    for( int i=0; i<text.length(); i++ ){
        if( !Character.isDigit( text.charAt( i ) ) ) return false;
    }
    return true;
    // Alternatively use regex: return text.matches("\\d+");
}
```

Alternatively, use a method to validate before reading.

```
public class Main {

    public static void main( String[] args ){
        if( args.length > 0 && containsInteger( args[0] ) ){
            int number = Integer.parseInt( args[0] );
        }
    }

...
```