Java

Key features:
• Write once, run anywhere
• Relatively fast
• Robust
• Secure
Java

General-purpose, high-level, object-oriented programming language
Java

General-purpose, high-level, object-oriented programming language

Key features:
• Write once, run anywhere
• Relatively fast
• Robust
• Secure
Step 1: Create/edit the program (e.g., Program.java)
Step 2: Compile the program
Step 3: Run the program
Repeat steps 1 – 3 until program output matches expected
Programming in Java

Step 1: Create/edit the program (eg, Program.java)

Step 2: Compile the program

Step 3: Run the program

Repeat steps 1 – 3 until program output matches expected
Step 1: Create/edit the program (eg, `Program.java`)

Step 2: Compile the program

```
> ~/workspace/dsaj/programs
$ 
```
Step 1: Create/edit the program (e.g., `Program.java`)

Step 2: Compile the program

```bash
> ~/workspace/dsaj/programs
$ javac -d out src/Program.java
```
Programming in Java

Step 1: Create/edit the program (eg, Program.java)

Step 2: Compile the program

```
> ~/workspace/dsaj/programs

$ javac -d out src/Program.java
$ 
```
Step 1: Create/edit the program (e.g., Program.java)

Step 2: Compile the program

```
> ~/workspace/dsaj/programs
$ javac -d out src/Program.java
$ 
```

Step 3: Run the program

```
> ~/workspace/dsaj/programs
$ 
```
Step 1: Create/edit the program (eg, Program.java)

Step 2: Compile the program

```bash
> ~/workspace/dsaj/programs
$ javac -d out src/Program.java
$ 
```

Step 3: Run the program

```bash
> ~/workspace/dsaj/programs
$ java Program
```
Programming in Java

Step 1: Create/edit the program (eg, Program.java)

Step 2: Compile the program

```
> ~/workspace/dsaj/programs
$ javac -d out src/Program.java
$ _
```

Step 3: Run the program

```
> ~/workspace/dsaj/programs
$ java Program
<program output>
$ _
```
Programming in Java

Step 1: Create/edit the program (eg, Program.java)

Step 2: Compile the program

```
> ~/workspace/dsaj/programs
$ javac -d out src/Program.java
$  
```

Step 3: Run the program

```
> ~/workspace/dsaj/programs
$ java Program
<program output>
$  
```

Repeat steps 1 – 3 until program output matches expected
HelloWorld.java

Standard output

the message "Hello, World"
| Standard output | the message “Hello, World” |
| Standard output | the message “Hello, World” |

`$ ~ /workspace/dsaj/programs`
The message "Hello, World" is written to standard output.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open HelloWorld.java</td>
<td></td>
</tr>
<tr>
<td>Standard output</td>
<td>the message “Hello, World”</td>
</tr>
</tbody>
</table>

```
$ javac -d out src/HelloWorld.java
$  
```
Hello, World

```
> ~/workspace/dsaj/programs

$ javac -d out src/HelloWorld.java
$ java HelloWorld
```

Standard output: the message "Hello, World"
| Standard output | the message “Hello, World” |

```
> ~/workspace/dsaj/programs
$ javac -d out src/HelloWorld.java
$ java HelloWorld
Hello, World
```

HelloWorld.java

// Writes the message "Hello, World" to standard output.

import stdlib.StdOut;

public class HelloWorld {
    // Entry point.
    public static void main(String[] args) {
        StdOut.println("Hello, World");
    }
}
// Writes the message "Hello, World" to standard output.

import stdlib.StdOut;

public class HelloWorld {
    // Entry point.
    public static void main(String[] args) {
        StdOut.println("Hello, World");
    }
}
Application Programming Interface (API)

Provides documentation for libraries and data types

Example:

```java
static void println(Object x)
prints an object and a newline to standard output

static void print(Object x)
prints an object to standard output
```
Application Programming Interface (API)

Provides documentation for libraries and data types

Example:

```
/♀st
stdlib.StdOut
static void println(Object x)
prints an object and a newline to standard output
static void print(Object x)
prints an object to standard output
```
Application Programming Interface (API)

Provides documentation for libraries and data types

Example:

| stdlib.StdOut |
|---|---|
| `static void println(Object x)` | prints an object and a newline to standard output |
| `static void print(Object x)` | prints an object to standard output |
Input and Output

Input types:
- Command-line input
- Standard input
- File input

Output types:
- Standard output
- File output
Input and Output

Input types:
• Command-line input
• Standard input
• File input

Output types:
• Standard output
• File output
Input and Output

Input types:
- Command-line input
- Standard input
- File input

Output types:
- Standard output
- File output
Input and Output

Input types:
- Command-line input
- Standard input
- File input

Output types:
- Standard output
- File output
Input and Output · Command-line Input

Command-line inputs (aka arguments) are strings listed next to the program name during execution. For example:

```
~/workspace/dsaj/programs
$ java Program input1 input2 input3 ...
```

The inputs are accessed within the entry-point (i.e., `public static void main(String[] args) { ... }`) function of the program as `args[0]`, `args[1]`, `args[2]`, ... .

Example:

```
~/workspace/dsaj/programs
$ java Program Galileo " Isaac Newton " Einstein
```

Command-line inputs (aka arguments) are strings listed next to the program name during execution.

Example:
```
$ java Program input1 input2 input3 ...
```

Command-line inputs (aka arguments) are strings listed next to the program name during execution:

```
> ~/workspace/dsaj/programs
$ java Program input1 input2 input3 ...
```

The inputs are accessed within the entry-point (ie, `public static void main(String[] args) { ... }`) function of the program as `args[0]`, `args[1]`, `args[2]`, ...
Input and Output · Command-line Input

Command-line inputs (aka arguments) are strings listed next to the program name during execution

```
media ~/workspace/dsaj/programs
$ java Program input1 input2 input3 ...
```

The inputs are accessed within the entry-point (ie, `public static void main(String[] args) { ... }`) function of the program as `args[0], args[1], args[2], ...`

Example:

```
media ~/workspace/dsaj/programs
$ java Program Galileo "Isaac Newton" Einstein
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Galileo&quot;</td>
<td>&quot;Isaac Newton&quot;</td>
<td>&quot;Einstein&quot;</td>
</tr>
</tbody>
</table>
Input and Output · Command-line Input

- UseArgument.java
- Command-line input: a name
- Standard output: a message containing the name
<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard output</td>
<td>a message containing the name</td>
</tr>
</tbody>
</table>
### Command-line Input

<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard output</td>
<td>a message containing the name</td>
</tr>
</tbody>
</table>

```bash
$ _
```

```bash
~$ /workspace/dsaj/programs
```
## Input and Output

- **Command-line Input**: UseArgument.java

  - Command-line input: a name

  - Standard output: a message containing the name

```
$ javac -d out src/UseArgument.java
```
### UseArgument.java

<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard output</td>
<td>a message containing the name</td>
</tr>
</tbody>
</table>

```
$ javac -d out src/UseArgument.java
$ 
```
## Input and Output

### Command-line Input

<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
</table>

| Standard output    | a message containing the name |

### Command Line

```bash
~/workspace/dsaj/programs
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
```
**Input and Output** - Command-line Input

<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard output</td>
<td>a message containing the name</td>
</tr>
</tbody>
</table>

```bash
$a ~/workspace/dsaj/programs
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
Hi, Alice. How are you?
$ _
```
### Command-line Input

Input:
```
UseArgument.java
```

- Command-line input: a name
- Standard output: a message containing the name

### Output

```
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
Hi, Alice. How are you?
$ java UseArgument Bob
```
### Command-line Input

<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard output</td>
<td>a message containing the name</td>
</tr>
</tbody>
</table>

```bash
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
Hi, Alice. How are you?
$ java UseArgument Bob
Hi, Bob. How are you?
$ _
```
**Input and Output**

- **Command-line Input**
  - UseArgument.java
  - Command-line input: a name
  - Standard output: a message containing the name

```bash
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
Hi, Alice. How are you?
$ java UseArgument Bob
Hi, Bob. How are you?
$ java UseArgument Carol
```
**Input and Output · Command-line Input**

<table>
<thead>
<tr>
<th>Command-line input</th>
<th>a name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard output</td>
<td>a message containing the name</td>
</tr>
</tbody>
</table>

```java
UseArgument.java
```

```bash
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
Hi, Alice. How are you?
$ java UseArgument Bob
Hi, Bob. How are you?
$ java UseArgument Carol
Hi, Carol. How are you?
$ _
```
Input and Output · Command-line Input

```
// Accepts a name as command-line argument and writes a message containing that name to standard output.
import stdlib.StdOut;
public class UseArgument {
    // Entry point.
    public static void main (String[] args) {
        StdOut.print("Hi,");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```
// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
Errors in a Program · Compile-time Errors

Compile-time errors are identified and reported by javac when it compiles a program.

Example:
```
// Accepts a name as command-line argument and writes a message containing that name to standard output.
import stdlib.StdOut;
public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```
Compile-time errors are identified and reported by `javac` when it compiles a program.
Compile-time errors are identified and reported by javac when it compiles a program.

Example:

```java
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi,");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```
Compile-time errors are identified and reported by javac when it compiles a program.

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(""). How are you");
    }
}
```

Errors in a Program · Compile-time Errors

Compile-time errors are identified and reported by javac when it compiles a program

Example:

```java
// Compile-time errors are identified and reported by javac when it compiles a program

// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```

$ javac -d out src/UseArgument.java
Compile-time errors are identified and reported by `javac` when it compiles a program.

**Example:**

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(" How are you?");
    }
}
```
Run-time errors are identified and reported by Java when it runs a program.

Example:

```
UseArgument.java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;
public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(" How are you?");
    }
}
```
Run-time errors are identified and reported by java when it runs a program.
Run-time errors are identified and reported by java when it runs a program.

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```
Errors in a Program · Run-time Errors

Run-time errors are identified and reported by java when it runs a program

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```

```
$ ~/workspace/dsaj/programs
```
Run-time errors are identified and reported by `java` when it runs a program.

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```

```
> ~/workspace/dsaj/programs
$ javac -d out src/UseArgument.java
```
Run-time errors are identified and reported by `java` when it runs a program.

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

class UseArgument {
    // Entry point.
    static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```

```
~/workspace/dsaj/programs
$ javac -d out src/UseArgument.java
$ 
```
Run-time errors are identified and reported by `java` when it runs a program

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
   public static void main(String[] args) {
      StdOut.print("Hi, ");
      StdOut.print(args[0]);
      StdOut.println(". How are you?");
   }
}
```

```bash
$ javac -d out src/UseArgument.java
$ java UseArgument
```
Run-time errors are identified and reported by `java` when it runs a program.

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.println(". How are you?");
    }
}
```

```bash
$ javac -d out src/UseArgument.java
$ java UseArgument
Hi, Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0
    at UseArgument.main(UseArgument.java:10)
$ _
```
Errors in a Program · Logic Errors

Logic errors are neither identified nor reported by `java`, but produce unintended output.

Example:

```
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;
public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.print(". How are you?");
    }
}
```
Logic errors are neither identified nor reported by java, but produce unintended output.
Errors in a Program · Logic Errors

Logic errors are neither identified nor reported by java, but produce unintended output.

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.print(" . How are you?");
    }
}
```
Logic errors are neither identified nor reported by Java, but produce unintended output.

Example:

```java
// UseArgument.java

// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.print(" . How are you? ");
    }
}
```

`~/workspace/dsaj/programs`

`$ `
Logic errors are neither identified nor reported by Java, but produce unintended output.

Example:

```java
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.print(\". How are you?\");
    }
}
```

```
$ javac -d out src/UseArgument.java
```
Logic errors are neither identified nor reported by `java`, but produce unintended output.

Example:

```java
// UseArgument.java

// Accepts a name as command-line argument; and writes a message containing that name to standard output.

import stdlib.StdOut;

public class UseArgument {
    // Entry point.
   public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.print(" . How are you?");
    }
}

$ javac -d out src/UseArgument.java
$ _
```
Logic errors are neither identified nor reported by java, but produce unintended output.

Example:

```java
// UseArgument.java

// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
    // Entry point.
    public static void main(String[] args) {
        StdOut.print("Hi, ");
        StdOut.print(args[0]);
        StdOut.print(" How are you?");
    }
}
```

```bash
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
```
Errors in a Program · Logic Errors

Logic errors are neither identified nor reported by java, but produce unintended output

Example:

```java
// Accepts a name as command-line argument; and writes a message containing that name to standard output.
import stdlib.StdOut;

public class UseArgument {
  // Entry point.
  public static void main(String[] args) {
    StdOut.print("Hi, ");
    StdOut.print(args[0]);
    StdOut.print(". How are you?");
  }
}
```

```
$ javac -d out src/UseArgument.java
$ java UseArgument Alice
Hi, Alice. How are you?
```