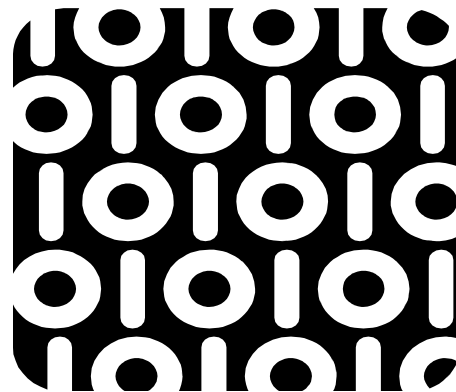


# Intro to Java Programming

## Getting Started

### ELEMENTS OF A JAVA PROGRAM

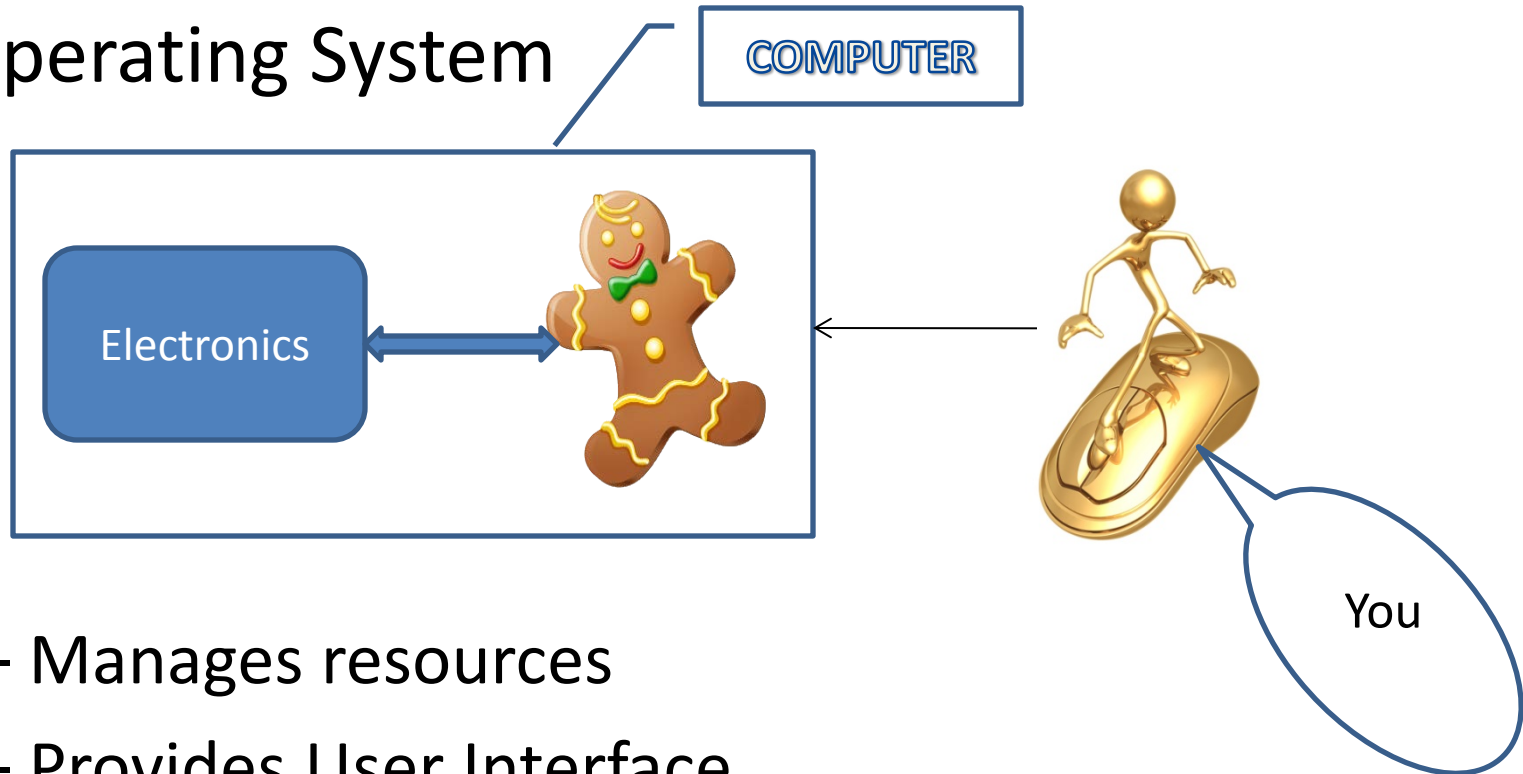


# Learning Objectives

- Discuss how a program runs
- Analyze elements of a Java program

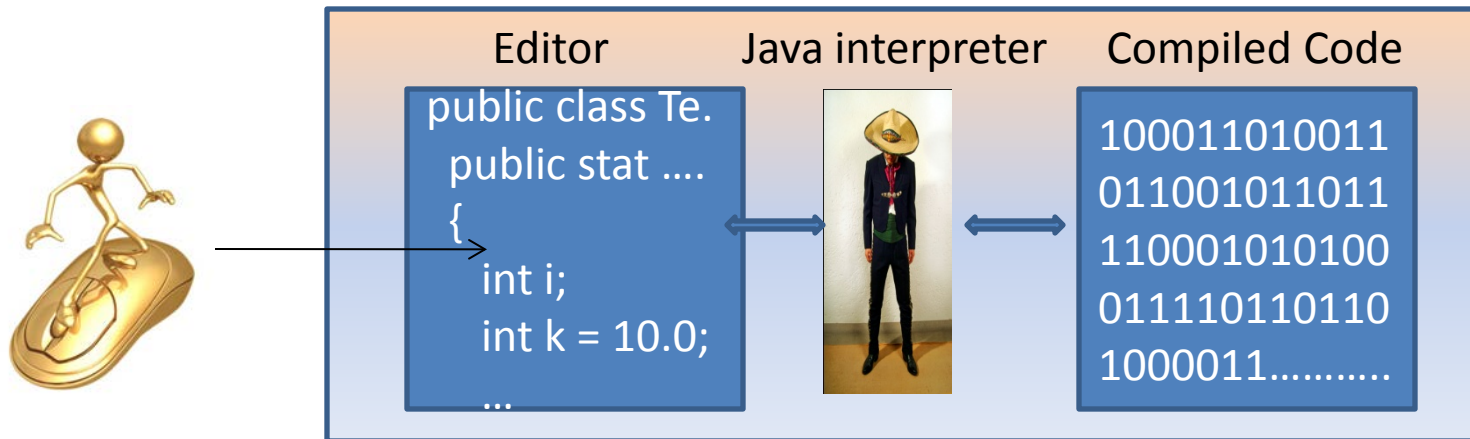
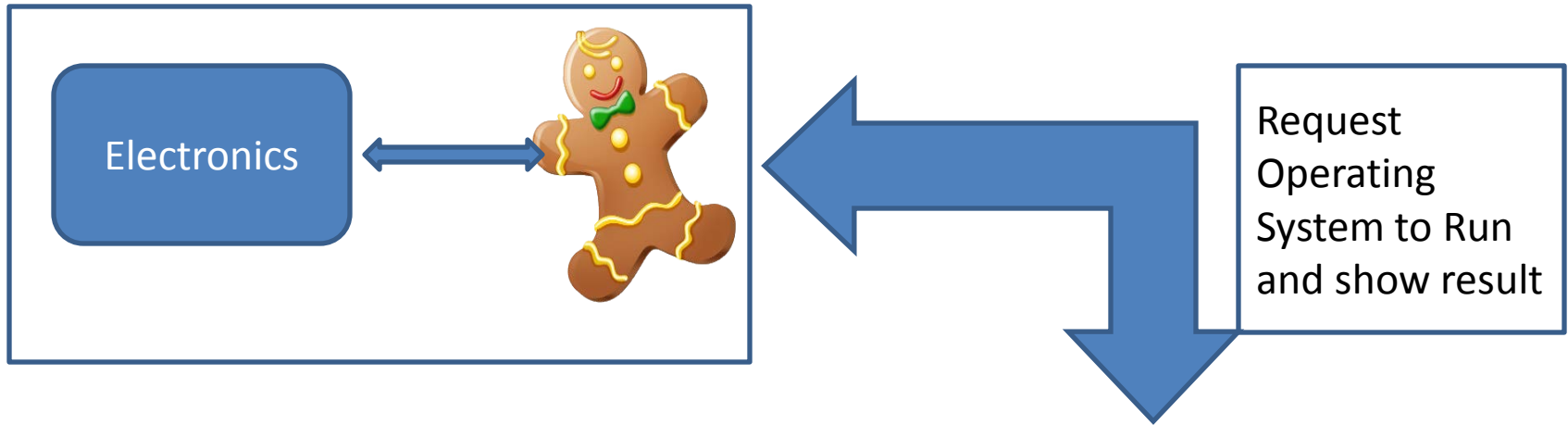
# Context of Programming

- Operating System



- Manages resources
- Provides User Interface
- Runs application software

# Context of Programming



# My First Program

```
// This is my first program
public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello World");
    }
}
```

# My First Program - Elements

- What are the elements of this program?

```
// This is my first program
```

- The line above is a comment
- You can write anything in that line after `//`
  - the program does not mind – called **inline** comment
- Used to communicate clarification, intention of use and other details to reader of the Source Code
- The ‘Computer System’ ignores comments
- **NOTE: This comment can be only one line physically**

```
// This is my first program
public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello world");
    }
}
```

# My First Program - Comments

- Is there an option for multiline comment?

`/*`

A multiline comments starts with a slash immediately followed by an Asterisk and can go for as many lines as desired. The comment then ends with Asterisk followed by a slash.

`*/`

- The program ignores multiline comment
- Used to communicate clarification, intention of use and other details to reader of the Source Code

# class

```
// This is my first program
public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello World");
    }
}
```

- This is the program
- **HelloWorld** is user defined
- Can use any name for the program
- Boundaries shown in **Dark Red**



# main

```
// This is my first program
public class HelloWorld {
    public static void main (String [] args) {
        System.out.println("Hello World");
    }
}
```

- This is where program starts
- Called the main method
- This line is always the same
- Boundaries shown in **Yellow**

# System.out.println


- Three predefined words
- System – name of a library
- out – standard output
- println – write a line
- Write a line shown between “....” within the parenthesis

```
// This is my first program
public class HelloWorld {
    public static void main (String [] args) {
        System.out.println("Hello World");
    }
}
```

# Compiling / Running

- The source code
  - File naming requirement
- Compilation
  - Binary File
- Debug and Run
- Does the program produce any output ?

# Run Time Error

- My program compiled without any error and I got an output 
  - Is your program done?
  - May be YES, and may be NO
- A program is not correct just because it produced a result (often called ‘output’)
- Output needs to be verified – more about it later

# println & print

- `println()` method is designed to insert a 'new line' after the line is written
- `print()` method can be used to write a line just like `println()`
- `print()` does not insert a 'new line' by default
- `System.out.print("Hello Universe");`
- Output: Hello Universe\_
- `System.out.print("\nHello Rover");`
- Output: Hello Rover ← This is a new line due to `\n`

—

# A few special characters

<code>\n</code>	Output a New line
<code>\t</code>	Output a Tab
<code>\"</code>	Output double quote sign
<code>'</code>	Output single quote sign
<code>\\</code>	Output backslash
<code>\r</code>	Advance the cursor to the beginning of the current line
<code>\b</code>	Output a backspace

`\` is called an Escape character

# A few other special characters

Character	Name	Description
{ ... }	Opening and closing braces	Denotes a block
( ... )	Opening and closing parenthesis	Used with methods
[ ... ]	Opening and closing brackets	Denotes an Array
//	Double slashes	Beginning of inline comment
" ... "	Opening and closing quotation marks	Encloses a string
;	Semicolon	Marks the end of a statement