Objectives

- Program Input and output
  - Console
- To represent characters using the char type
- To represent a string using the String type
- To distinguish syntax errors, runtime errors, and logic errors and debug errors

Chapter 2

Elementary Programming

Displaying Text on the console

- Java Library provide the following methods to display information on the console:
  - System.out.println
  - System.out.print
- The arguments can be any expression, including a variable, literal, or value returned by a method.
- The prinltln method always advances to the next line after displaying its arguments, the print method does not.

Escape Sequences for Special Characters

Description	*Escape Sequence*	*Unicode*
Backspace	`\b`	`\u0008`
Tab	`\t`	`\u0009`
Linefeed	`\n`	`\u000A`
Carriage return	`\r`	`\u000D`
Backslash	`\\`	`\u005C`
Single Quote	`\'`	`\u0027`
Double Quote	`\"`	`\u0022`

println examples

- System.out.println(“He yelled \”Stop! \” and we stopped.”);
  output: He Yelled “Stop!” and we stopped.

- double temp = 25.5;
- System.out.println(“The temperature is: “ + temp);
  output: The temperature is: 25.5

- System.out.println(“Hello ” + 2 + “there!”);
  output: Hello 2there!

- System.out.println(“PLP\\360\nVancouver”);
  output: PLP\360
  Vancouver

Reading Numbers from the Keyboard

```
Scanner input = new Scanner(System.in);
int value = input.nextInt();
```

Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.

Sample program using Scanner

```
import java.util.Scanner;
public class Lottery {
    public static void main(String[] args) {
        int winningNumber;
        double ticket_price;

        Scanner input = new Scanner(System.in);
        System.out.println("Please enter the Lottery number:");
        winningNumber = input.nextInt()

       System.out.print("\n\nPlease enter the ticket_price: ");
       ticket_price = input.nextDouble();
       .
        .
  }
}
```

Character Data Type

char letter = 'A'; (ASCII)

char numChar = '4'; (ASCII)

NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

```
char ch = 'a';
System.out.println(++ch);
```

ASCII Character Set

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.2 ASCII Character Set in the Hexadecimal Index

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	“	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del

The String Type

The char type only represents one character. To represent a string of characters, use the data type called String. For example,

```
String message = "Welcome to Java";
```

String is actually a predefined class in the Java library just like the System class and JOptionPane class. The String type is not a primitive type. It is known as a *reference type*. Any Java class can be used as a reference type for a variable. Reference data types will be thoroughly discussed in Chapter 7, “Objects and Classes.” For the time being, you just need to know how to declare a String variable, how to assign a string to the variable, and how to concatenate strings.

String Concatenation

```
// Three strings are concatenated
String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2
String s = "Chapter" + 2; // s becomes Chapter2

// String Supplement is concatenated with character
B
String s1 = "Supplement" + 'B'; // s1 becomes
SupplementB
```

Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

Syntax Errors

```
public class ShowSyntaxErrors {
  public static main(String[] args) {
    System.out.println("Welcome to Java);
  }
}
```

Runtime Errors

```
public class ShowRuntimeErrors {
  public static void main(String[] args) {
    System.out.println(1 / 0);
  }
}
```

Logic Errors

```
public class ShowLogicErrors {
  public static void main(String[] args) {
    System.out.println("Celsius 35 is Fahrenheit degree ");
    System.out.println((9 / 5) * 35 + 32);
  }
}
```

Debugging

- Logic errors are called *bugs*.
- The process of finding and correcting errors is called debugging.
- A common approach to debugging is to use a combination of methods to narrow down to the part of the program where the bug is located.
- You can hand-trace the program (i.e., catch errors by reading the program), or you can insert print statements in order to show the values of the variables or the execution flow of the program.
- This approach might work for a short, simple program. But for a large, complex program, the most effective approach for debugging is to use a debugger utility.

Preparation

- Quiz on Wednesday
- Read Chapter 1 and 2
- Review the slides