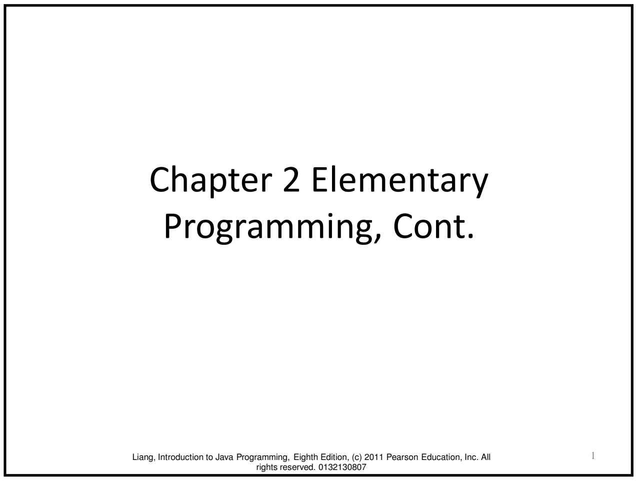
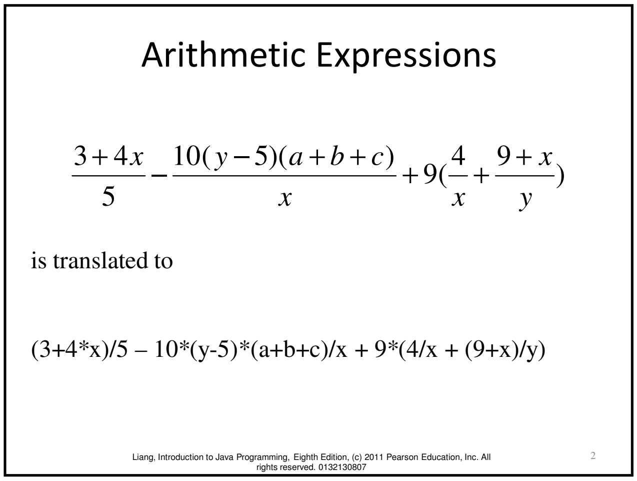


Chapter 2 Elementary Programming, Cont.

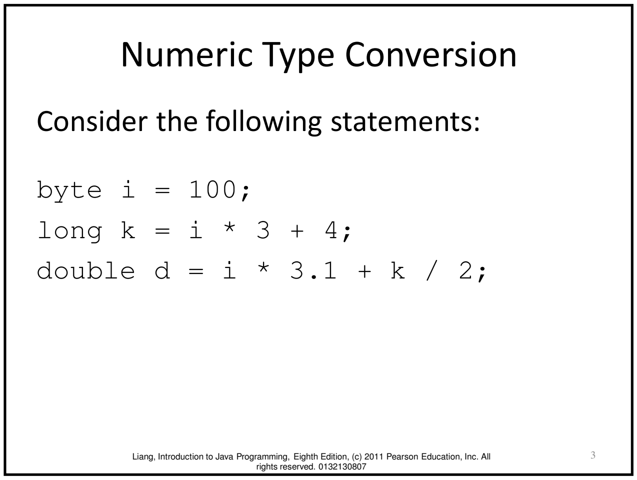


Arithmetic Expressions

$$\frac{3+4x}{5}-\frac{10(y-5)(a+b+c)}{x}+9(\frac{4}{x}+\frac{9+x}{y})$$

is translated to

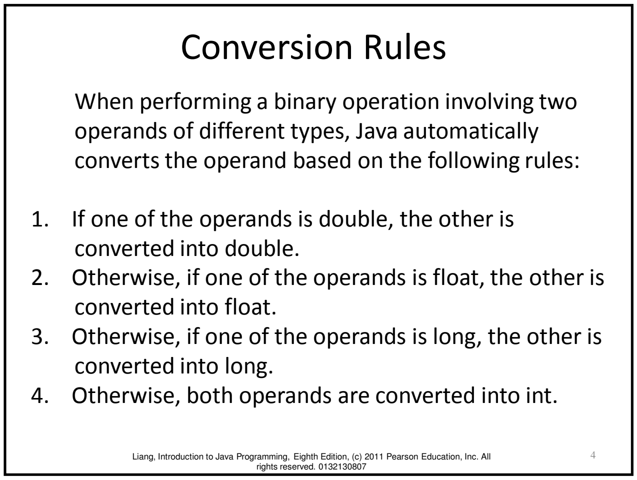
(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)



Numeric Type Conversion

Consider the following statements:

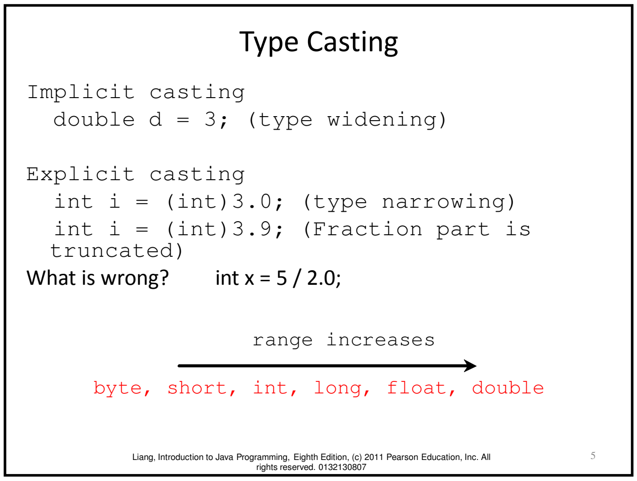
```
byte i = 100;
long k = i * 3 + 4;
double d = i * 3.1 + k / 2;
```



Conversion Rules

When performing a binary operation involving two operands of different types, Java automatically converts the operand based on the following rules:

1. If one of the operands is double, the other is converted into double.
2. Otherwise, if one of the operands is float, the other is converted into float.
3. Otherwise, if one of the operands is long, the other is converted into long.
4. Otherwise, both operands are converted into int.

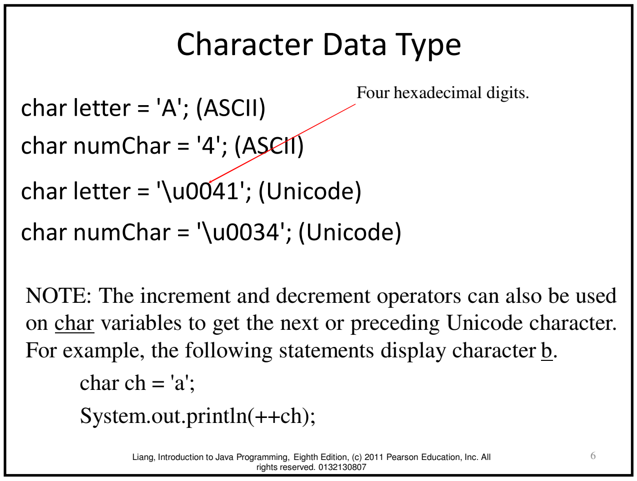


Type Casting

```
Implicit casting
  double d = 3; (type widening)

Explicit casting
  int i = (int)3.0; (type narrowing)
  int i = (int)3.9; (Fraction part is
 truncated)
```

What is wrong?   int x = 5 / 2.0;

```
          range increases
     ───────────────────────>
byte, short, int, long, float, double
```



Character Data Type

Four hexadecimal digits.

char letter = 'A'; (ASCII)

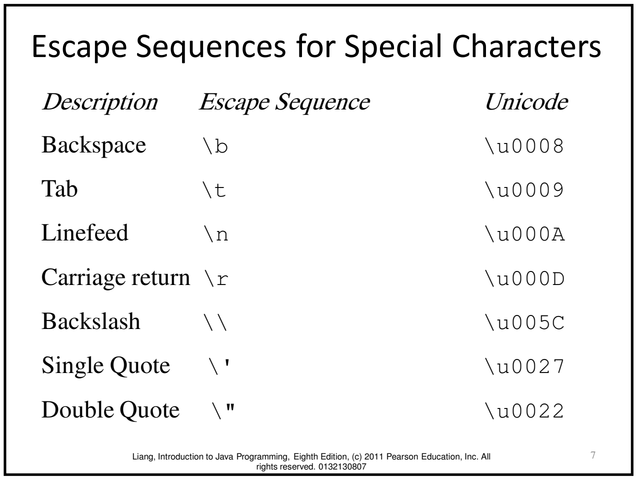
char numChar = '4'; (ASCII)

char letter = '\u0041'; (Unicode)

char numChar = '\u0034'; (Unicode)

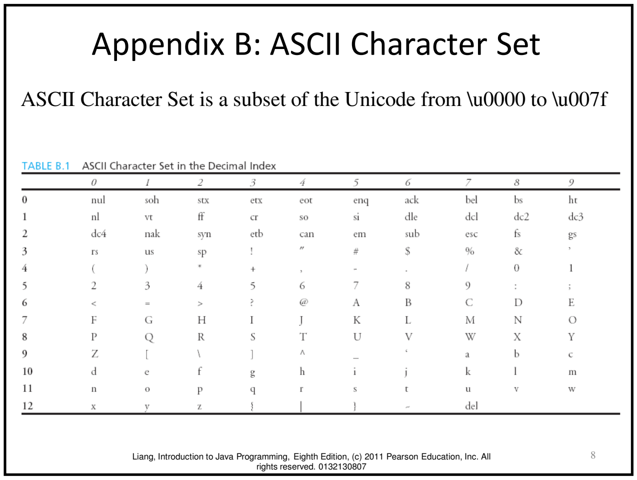
NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

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



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`



Appendix B: ASCII Character Set

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

TABLE B.1 ASCII Character Set in the Decimal Index

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

ASCII Character Set, cont.

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

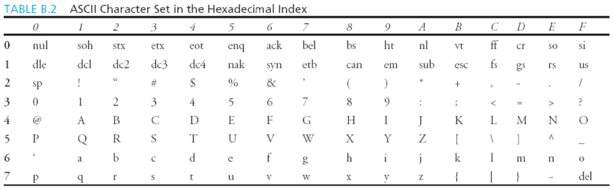


TABLE B.2 ASCII Character Set in the Hexadecimal Index

	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

Casting between char and Numeric Types

```
int i = 'a'; // Same as int i = (int)'a';

char c = 97; // Same as char c = (char)97;
```

The String Type

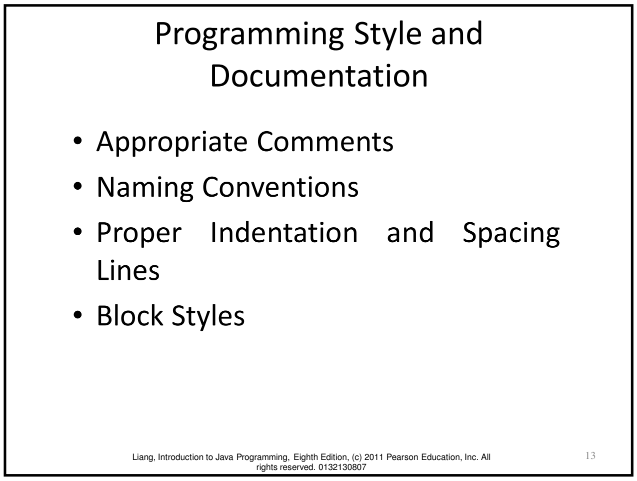
The char type only represents one character. To represent a string of characters, use the data type called String. For example,

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.

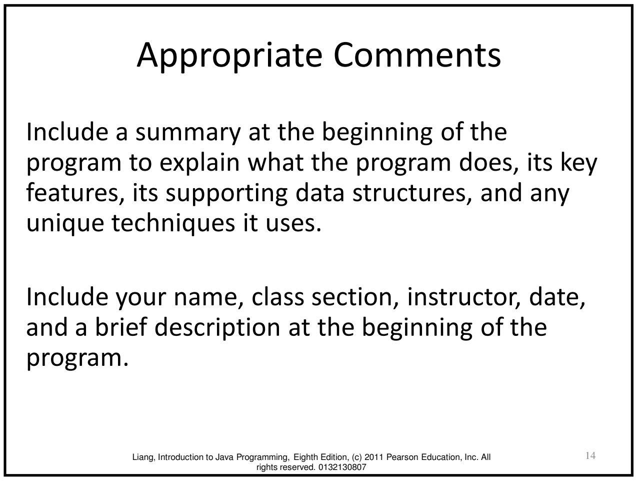
String Concatenation

```
// 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 Style and Documentation

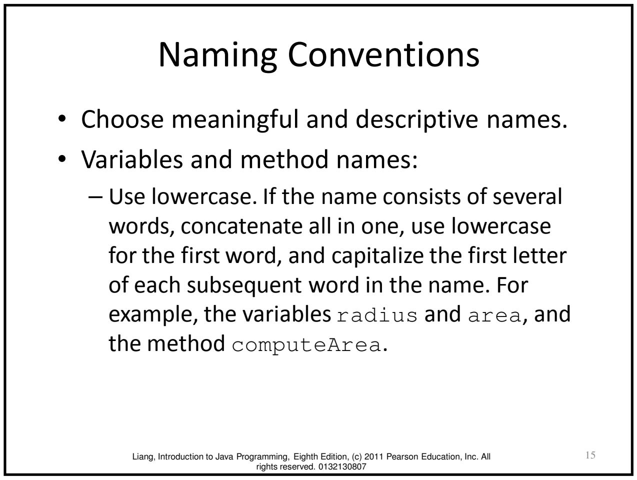
- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles



Appropriate Comments

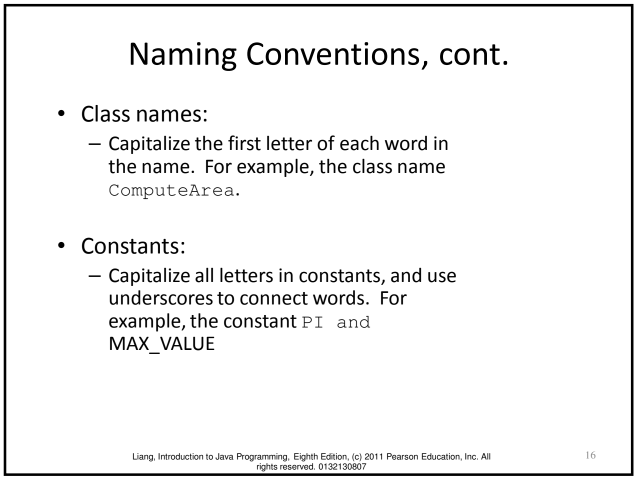
Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.



Naming Conventions

- Choose meaningful and descriptive names.
- Variables and method names:
  - Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables `radius` and `area`, and the method `computeArea`.



Naming Conventions, cont.

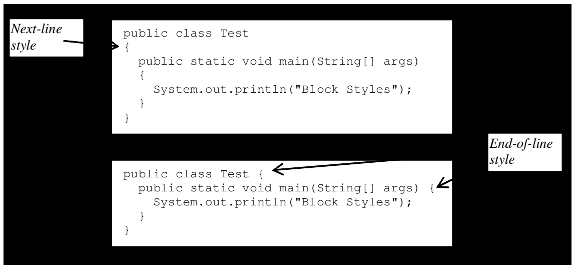
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeArea`.
- Constants:
  - Capitalize all letters in constants, and use underscores to connect words. For example, the constant `PI and` MAX_VALUE

Proper Indentation and Spacing

- Indentation
  - – Indent two spaces.
- Spacing
  - – Use blank line to separate segments of the code.

Block Styles

Use end-of-line style for braces.



```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

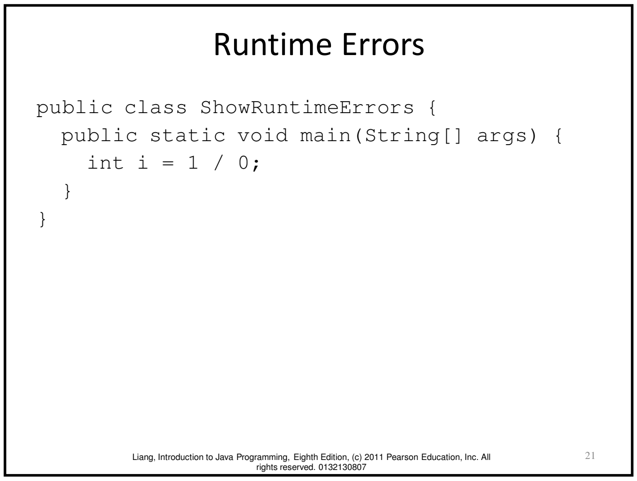
```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Programming Errors

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

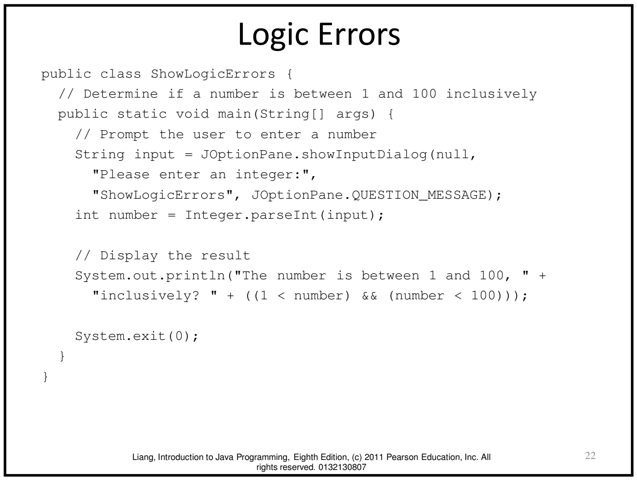
Syntax Errors

```
public class ShowSyntaxErrors {
  public static void main(String[] args) {
    i = 30;
    System.out.println(i + 4);
  }
}
```



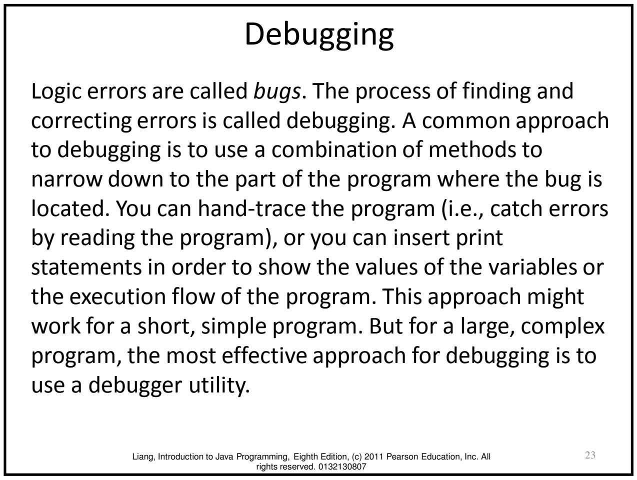
Runtime Errors

```
public class ShowRuntimeErrors {
  public static void main(String[] args) {
    int i = 1 / 0;
  }
}
```



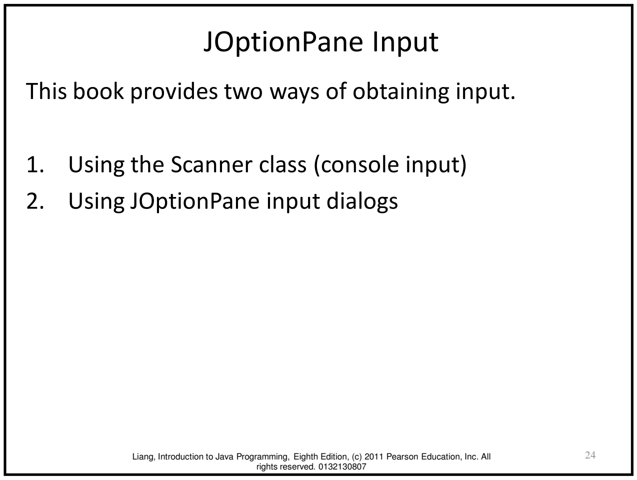
Logic Errors

```
public class ShowLogicErrors {
  // Determine if a number is between 1 and 100 inclusively
  public static void main(String[] args) {
    // Prompt the user to enter a number
    String input = JOptionPane.showInputDialog(null,
      "Please enter an integer:",
      "ShowLogicErrors", JOptionPane.QUESTION_MESSAGE);
    int number = Integer.parseInt(input);

    // Display the result
    System.out.println("The number is between 1 and 100, " +
      "inclusively? " + ((1 < number) && (number < 100)));

    System.exit(0);
  }
}
```



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.



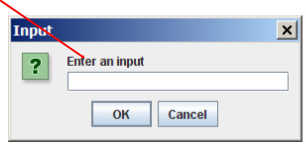
JOptionPane Input

This book provides two ways of obtaining input.

1. Using the Scanner class (console input)
2. Using JOptionPane input dialogs

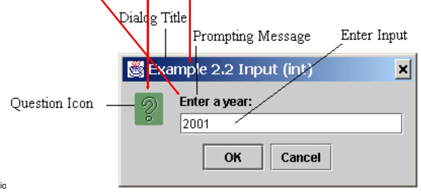
Getting Input from Input Dialog Boxes

```
String input =  JOptionPane.showInputDialog(
  "Enter an input");
```



Getting Input from Input Dialog Boxes

```
String string = JOptionPane.showInputDialog(
   null, “Prompting Message”,  “Dialog Title”,
   JOptionPane.QUESTION_MESSAGE);
```



Converting Strings to Integers

The input returned from the input dialog box is a string. If you enter a numeric value such as 123, it returns “123”. To obtain the input as a number, you have to convert a string into a number.

To convert a string into an int value, you can use the static parseInt method in the Integer class as follows:

```
int intValue = Integer.parseInt(intString);
```

where intString is a numeric string such as “123”.

Converting Strings to Doubles

To convert a string into a double value, you can use the static parseDouble method in the Double class as follows:

```
double doubleValue =Double.parseDouble(doubleString);
```

where doubleString is a numeric string such as “123.45”.