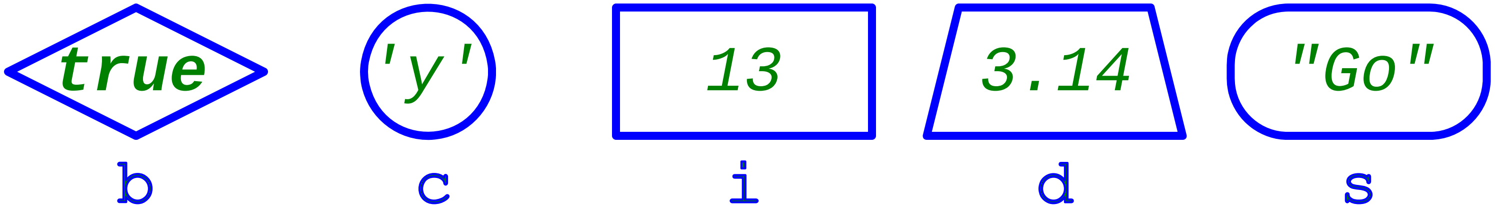
Variables, Types, Values

Variables

- A ***variable*** is the name of a “location” that “stores” a ***value*** of a particular ***type***
  - – We might say the variable “has” that value
  - – We might say the variable “has” that type or “is of” that type

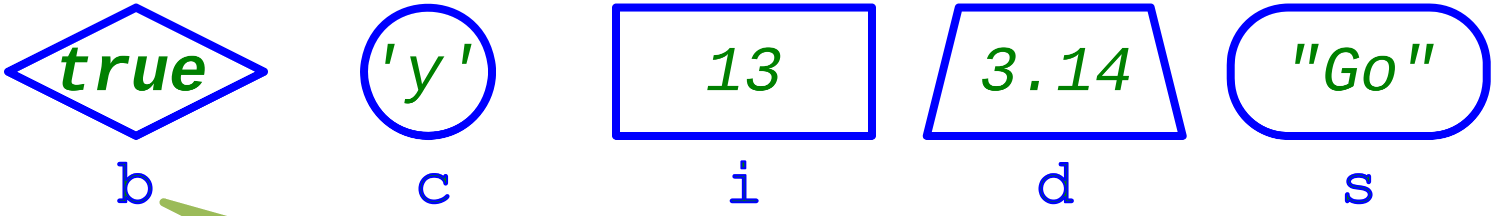
Examples



`true` `'y'` `13` `3.14` `"Go"`

`b` `c` `i` `d` `s`

Examples



This is a **boolean** variable

b

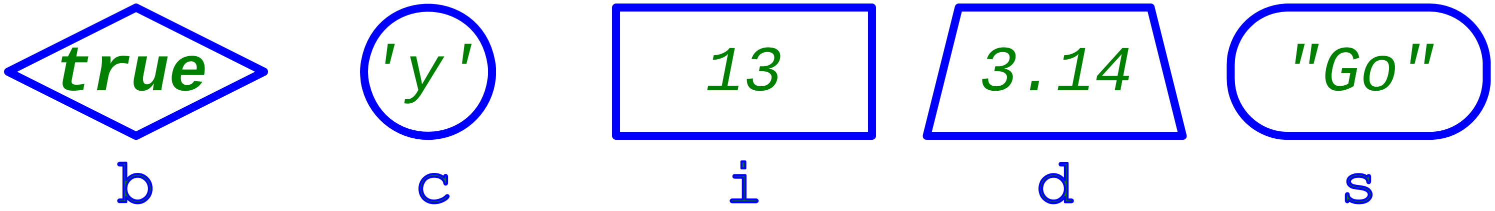
whose value is ***true***, i.e.,

*b = **true***

or, more simply, just

b

Examples



true

b

'y'

c

13

i

3.14

d

"Go"

s

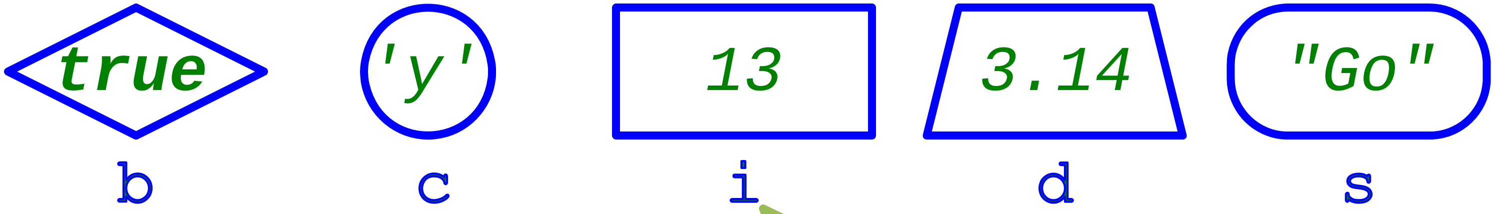
This is a **char** variable

c

whose value is 'y', i.e.,

c = 'y'

Examples



true 'y' 13 3.14 "Go"

b c i d s

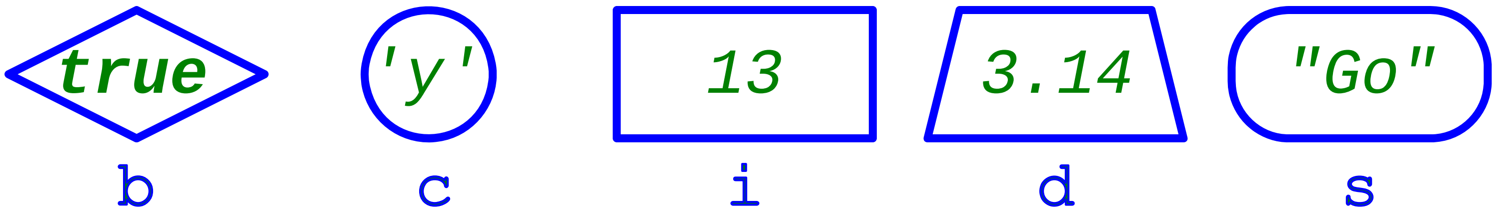
This is a **int** variable

i

whose value is *13*, i.e.,

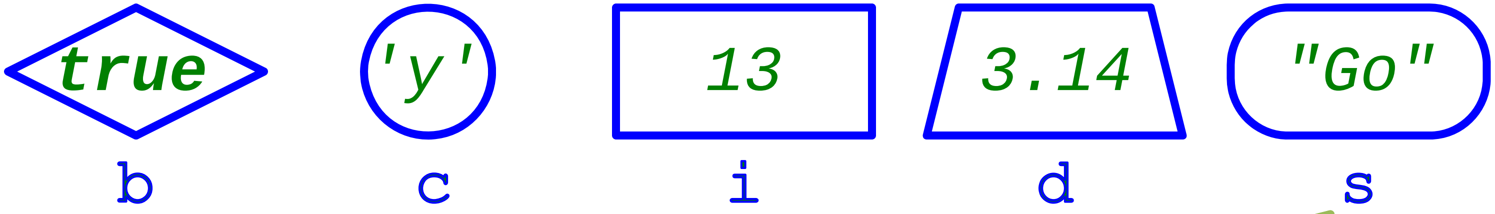
$i = 13$

Examples



This is a **double** variable
d
whose value is *3.14*, i.e.,
d = 3.14

Examples



This is a `String` variable `s` whose value is `"Go"`, i.e.,

`s = "Go"`

(or `s = <'G', 'o'>`)

Types

- A ***type*** is the name of the set of all possible values that a variable might have
- Examples:
  - A variable of type `String` might have values like *`"foo"`*, *`"Hello World"`*, etc.
  - A variable of type **`int`** might have values like *`-1`*, *`18`*, etc.
  - A variable of type **`double`** might have values like *`3.1416`*, *`10.0`*, etc.

Program vs. Mathematical Variables

- A ***program variable*** has a particular value at any one time during program execution, and that value (generally) may change at other times
- A ***mathematical variable*** stands for an arbitrary but fixed value

Program vs. Mathematical Types

- A ***program type*** has a corresponding ***mathematical type*** that ***models*** it

Program vs. Mathematical Types

- A ***program type*** has a corresponding ***mathematical type*** that ***models*** it

When reasoning about a *program variable* of a given *program type*, treat its value at any given time as if it were a *mathematical variable* of the corresponding *mathematical type*.

Mathematical Models

Program type	***Mathematical type***
`String`	***string of character***
`boolean`	***boolean***
`char`	***character***
`int`	***integer*** (*-2147483648* through *2147483647*)
`double`	***real*** (about $\pm 10^{\pm 308}$, 15 significant digits)

Mathematical Models

Program type	***Mathematical type***
`String`	***string of character***
`boolean`	
`char`	
`int`	ugh
`double`	(about ±$10^{\pm308}$, 15 significant digits)

`String` is ***built-in*** to Java; **`boolean`**, **`char`**, **`int`**, and **`double`** are among the 8 ***primitive*** (and also built-in) types of Java; differences later.

Mathematical Models

Program type	***Mathematical type***
String	***string of character***
	boolean
	character
	integer (*-2147483648* through *2147483647*)
double	***real*** (about $\pm 10^{\pm 308}$, 15 significant digits)

All these mathematical types are “built-in” to mathematics!

Mathematical Models

Program code is shown in a `blue fixed-width` font, with keywords in **`bold`**.

Program type	
`String`	***string of character***
`boolean`	***boolean***
`char`	***character***
`int`	***integer*** (*-2147483648* through *2147483647*)
`double`	***real*** (about $\pm 10^{\pm 308}$, 15 significant digits)

Mathematical Models

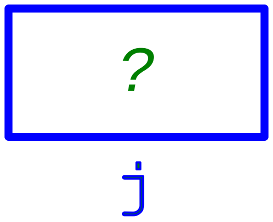
Mathematics is shown in a *green fixed-width italic* font, with keywords in ***bold***.

	Mathematical type
String	***string of character***
boolean	***boolean***
char	***character***
int	***integer*** (*-2147483648* through *2147483647*)
double	***real*** (about $\pm10^{\pm308}$, 15 significant digits)

Declaring a Variable

- When you ***declare*** a program variable, you both provide a name for a location to store its value, and indicate its program type
  - Recall: the program type determines the mathematical type, which in turn determines the possible values the variable can have

```
int j;
```



Declaring a Variable

- When you ***declare*** a program variable, you both provide a name for a location to store its value, and indicate its program type
  - Recall: the pro
    mathematical
    the possible v

```
int j;
```

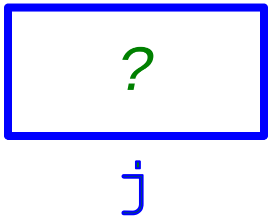
The standard Java convention for naming variables is to use ***camel case***: start with a lower case letter and only capitalize the first letter of each following word, e.g., `myLuckyNumber`

Declaring a Variable

- When you ***declare*** a program variable, you both provide a name for a location to store its value, and indicate its program type
  - Recall: the rmines the
    mathematic rn determines
    the possible e can have

```
int j;
```

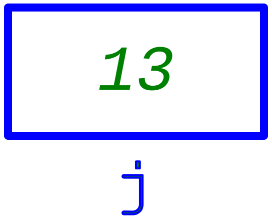
This is an **int** variable j whose value is ***undefined***.



Initializing a Variable

- To ***initialize*** a variable, you ***assign*** it a ***value***
  - Recall: the program type determines the mathematical type, which in turn determines the possible values the variable can have

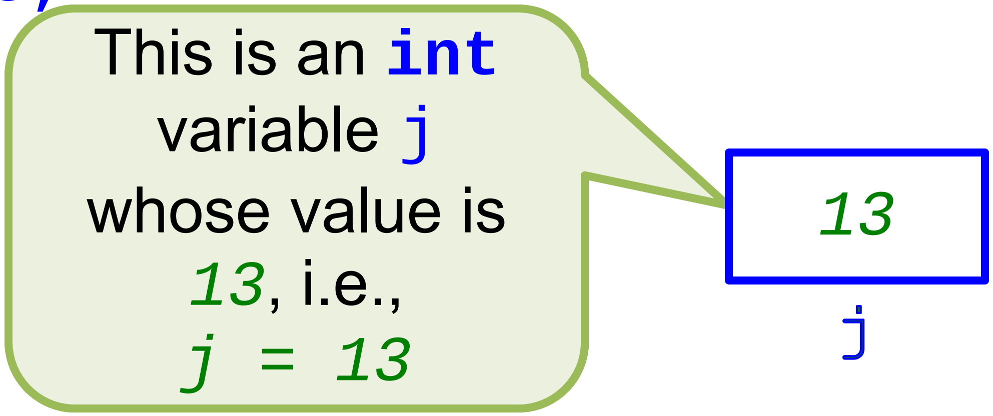
```
int j = 13;
```



Initializing a Variable

- To ***initialize*** a variable, you ***assign*** it a ***value***
  - Recall: the program type determines the mathematical type, which in turn determines the possible values the variable can have

```
int j = 13;
```



This is an **`int`** variable `j` whose value is *13*, i.e., *j = 13*

Reasoning: Tracing Tables

Code	***State***
	x = 1.414
`int j = 13;`	
	x = 1.414 *j = 13*

Reasoning: Tracing Tables

Code	***State***
`int j = 13`	
	x = 1.414 *j = 13*

Every other row in the left column contains some ***program*** statement(s).

Reasoning: Tracing Tables

Code	***State***
`int j = 13`	

	`x = 1.414`
	`j = 13`

Every other row in the right column contains some ***mathematical*** sentences ("facts").

Reasoning: Tracing Tables

Coc	***tate***
`int j = 13;`	
	x = 1.414 *j = 13*

This equal sign, in code, means ***assignment*** of a value to a program variable.

Reasoning: Tracing Tables

This equal sign, in mathematics, means ***equality*** of two values.

Coc	tate
`int j = 13;`	
	x = 1.414 *j = 13*

Reasoning: Tracing Tables

There is no value for mathematical variable *j* in this state because program variable `j` hasn’t been declared yet.

	State
	x = 1.414
`int j = 13;`	
	x = 1.414 *j = 13*

Reasoning: Tracing Tables

	State
	x = 1.414
`int j = 13;`	
	x = 1.414 *j = 13*

There is a value for *j* in this state because `j` has been declared before this state.

Literals

- A data value appearing, literally, in a program is called a ***literal***

```java
String fileName = "foo.txt";
boolean found = false;
char win = 'W';
int j = 13;
double ht = 9.27;
```

Literals

- A data value appearing, literally, in a program is called a ***literal***

```
String fileName = "foo.txt";
boolean found = false;
char win = 'W';
int j = 13;
double ht = 9.27;
```

This is a `String` literal; written as characters between double-quote marks: `"…"`

Literals

- A data value appearing, literally, in a program is called a ***literal***

```
String fileName = "foo.txt";
boolean found = false;
char win = 'W';
int j = 13;
double ht = 9.27;
```

This is a **`boolean`** literal; must be either **`true`** or **`false`**.

Literals

- A data value appearing, literally, in a program is called a ***literal***

```
String fileName = "foo.txt";
boolean found = false;
char win = 'W';
int j = 13;
double ht = 9.27;
```

This is a `char` literal; normally written as a single character between single-quote marks: `'…'`

Literals

- A data value appearing, literally, in a program is called a ***literal***

```
String fileName = "f
boolean found = fals
char win = 'W';
int j = 13;
double ht = 9.27;
```

This is an **`int`** literal; normally written (as in mathematics) as a decimal constant.

Literals

- A data value appearing, literally, in a program is called a ***literal***

```
String fileName = "foo.txt";
boolean found = fals
char win = 'W';
int j = 13;
double ht = 9.27;
```

This is a **`double`** literal; normally written (as in mathematics) as a decimal constant *with* a decimal point.

Forms of Literals

Program type	***Literal examples***
`String`	`"I\'m"` `"at OSU"`
`boolean`	**`true`** **`false`**

`char`	`'A'` `'\t'` `'\"'`
	`'\u03c0'`

`int`	`29` `-13`
	`035` `0x1a`

`double`	`18.` `18.0`
	`8E-4` `6.023E23`

Forms of Literals

Program type	***Literal examples***
`String`	`"I\'m"` `"at OSU"`
`boolean`	**`true`** **`false`**

`c`	`A'` `'\t'` `'\"'`
	`'\u03c0'`

`int`	`29` `-13`
	`035` `0x1a`

`double`	`18.` `18.0`
	`8E-4` `6.023E23`

escaped special character: single-quote

Forms of Literals

Program type	***Literal examples***
`String`	`"I\'m"` `"at OSU"`
`bo`	**`true`** **`false`**

`char`	`'A'` `'\t'` `'\"'`
	`'\u03c0'`

`int`	`29` `-13`
	`035` `0x1a`

`double`	`18.` `18.0`
	`8E-4` `6.023E23`

non-printing character: tab

Forms of Literals

Program type	***Literal examples***
St	I\'m" "at OSU"
bo	**true** **false**
char	'A' '\t' '\"' '\u03c0'
int	29 -13 035 0x1a
double	18. 18.0 8E-4 6.023E23

Unicode character: small Greek π

Forms of Literals

Program type	*Literal examples*
St	I\'m" "at OSU"
bo	**true** **false**
char	'A' '\t' '\"' '\u03c0'
int	29 -13 035 0x1a
double	18. 18.0 8E-4 6.023E23

octal integer (base-8): 29 in decimal

Forms of Literals

Program type	***Literal examples***
`St`	`I\'m"     "at OSU"`
`bo`	**`true    false`**

`char`	`'A'    '\t'    '\"'`
	`'\u03c0'`

`int`	`29    -13`
	`035    0x1a`

`double`	`18.    18.0`
	`8E-4    6.023E23`

hexadecimal
integer (base-16):
26 in decimal

Forms of Literals

Program type	***Literal examples***
`String`	`"I\'m"` `"at OSU"`
`bo`	**`true`** **`false`**

	`'A'` `'\t'` `'\"'`
	`'\u03c0'`

`int`	`29` `-13`
	`035` `0x1a`

`double`	`18.` `18.0`
	`8E-4` `6.023E23`

scientific notation:

8 x 10^{-4}

Constants

- A variable whose value is initialized and never changed is called a ***constant***

```
int myLuckyNumber = 13;
double avogadro = 6.023E23;
```

Constants

- A variable whose value is initialized and never changed is called a ***constant***

```
final int myLuckyNumber = 13;
final double avogadro = 6.023E23;
```

The keyword `final` indicates to the compiler your intent that a variable is actually a constant.

Constants

- A variable whose value is initialized and never changed is called a ***constant***

```
final int MY_LUCKY_NUMBER = 13;
final double AVOGADRO = 6.023E23;
```

The standard Java convention for naming constants is to use all upper case letters and to separate the words with '_'.

Resources

- *Big Java Late Objects*, Chapter 2
  - http://proquest.safaribooksonline.com.proxy.lib.ohio-state.edu/book/programming/java/9781118087886/chapter-2-fundamental-data-types/navpoint-18