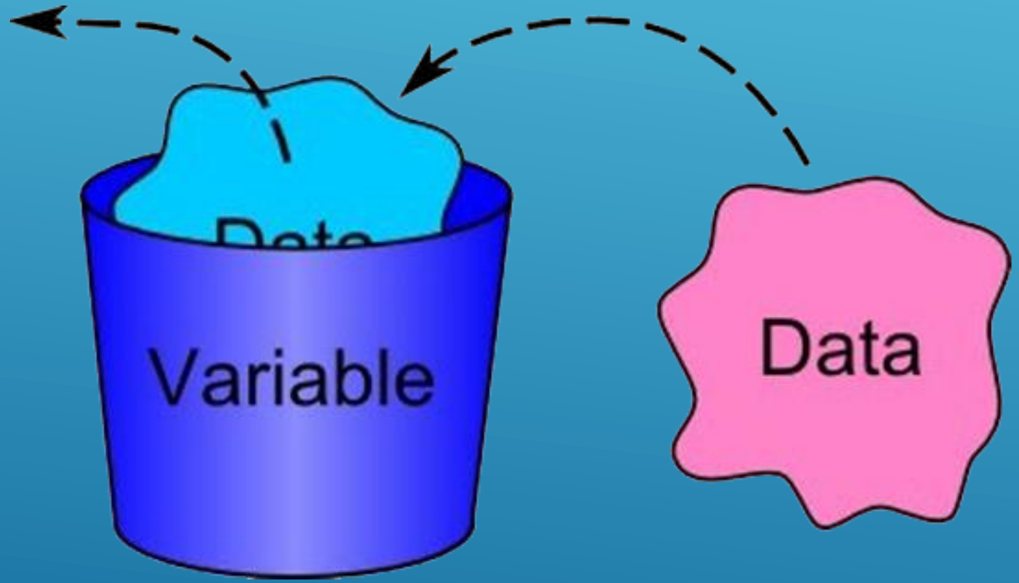
VARIABLES/DATATYPES.

WHAT IS VARIABLE?

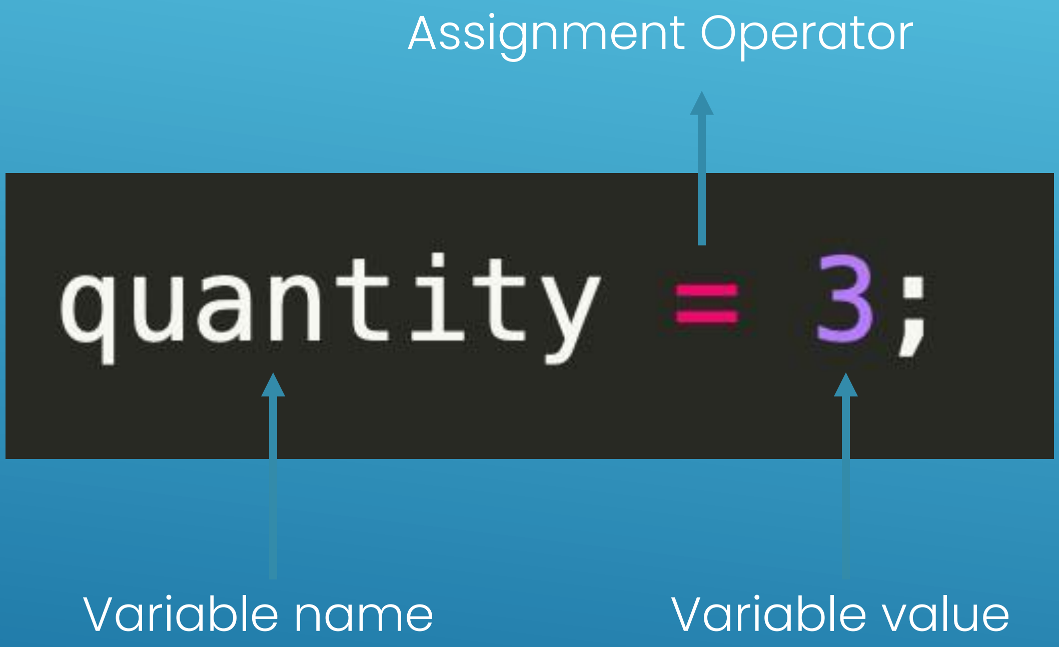


SELECTING THE PROPER DATA TYPE IS IMPORTANT BECAUSE A VARIABLE'S DATA TYPE DETERMINES THE AMOUNT OF MEMORY THE VARIABLE USES, AND THE WAY THE VARIABLE FORMATS AND STORES DATA.

HOW TO DECLARE VARIABLES?

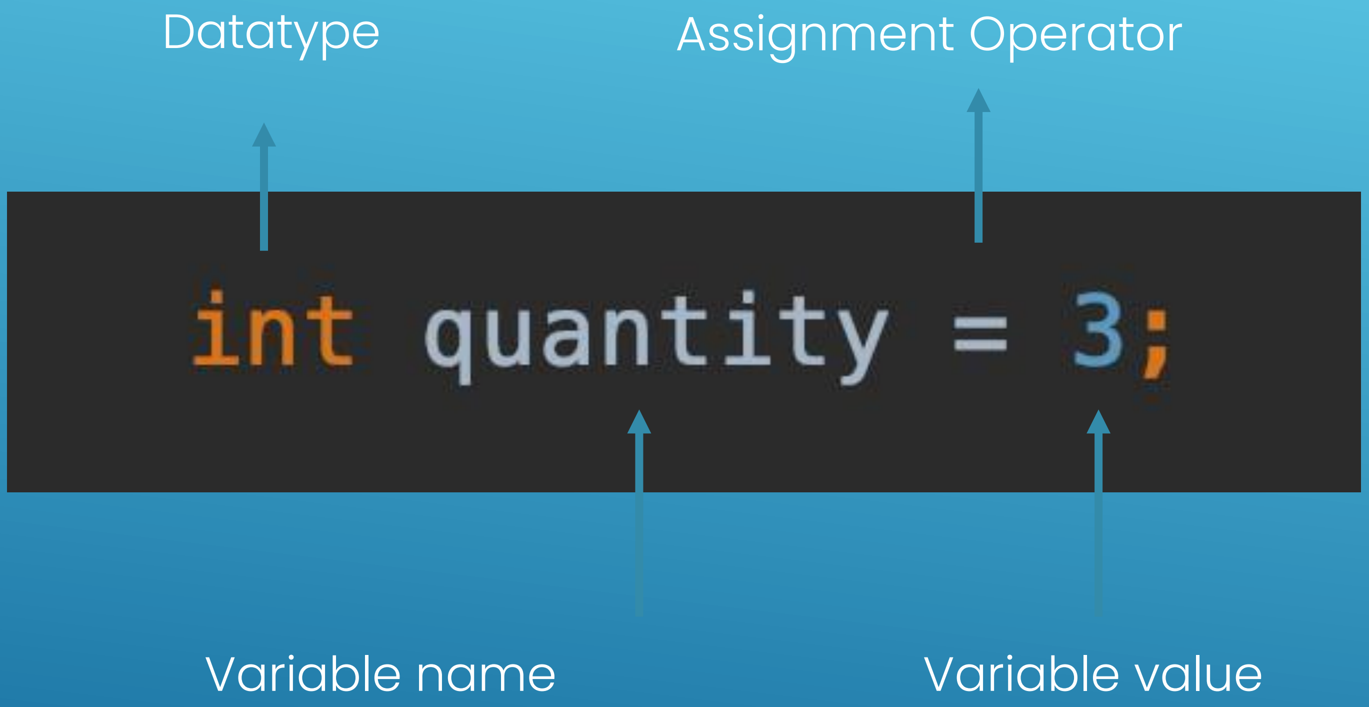
```
DataType VariableName;
```

HOW TO ASSIGN VARIABLE A VALUE?



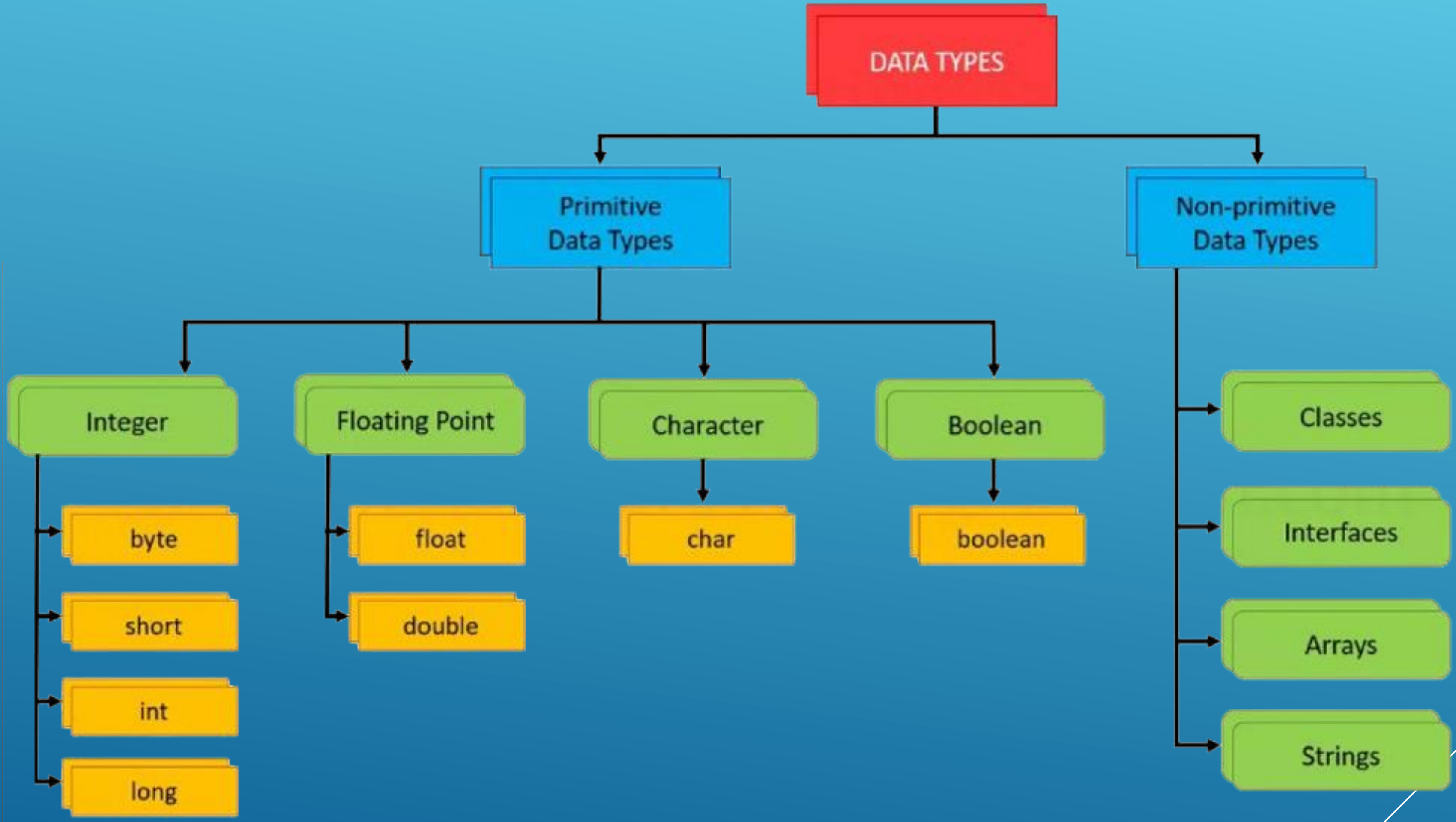
```
quantity = 3;
```

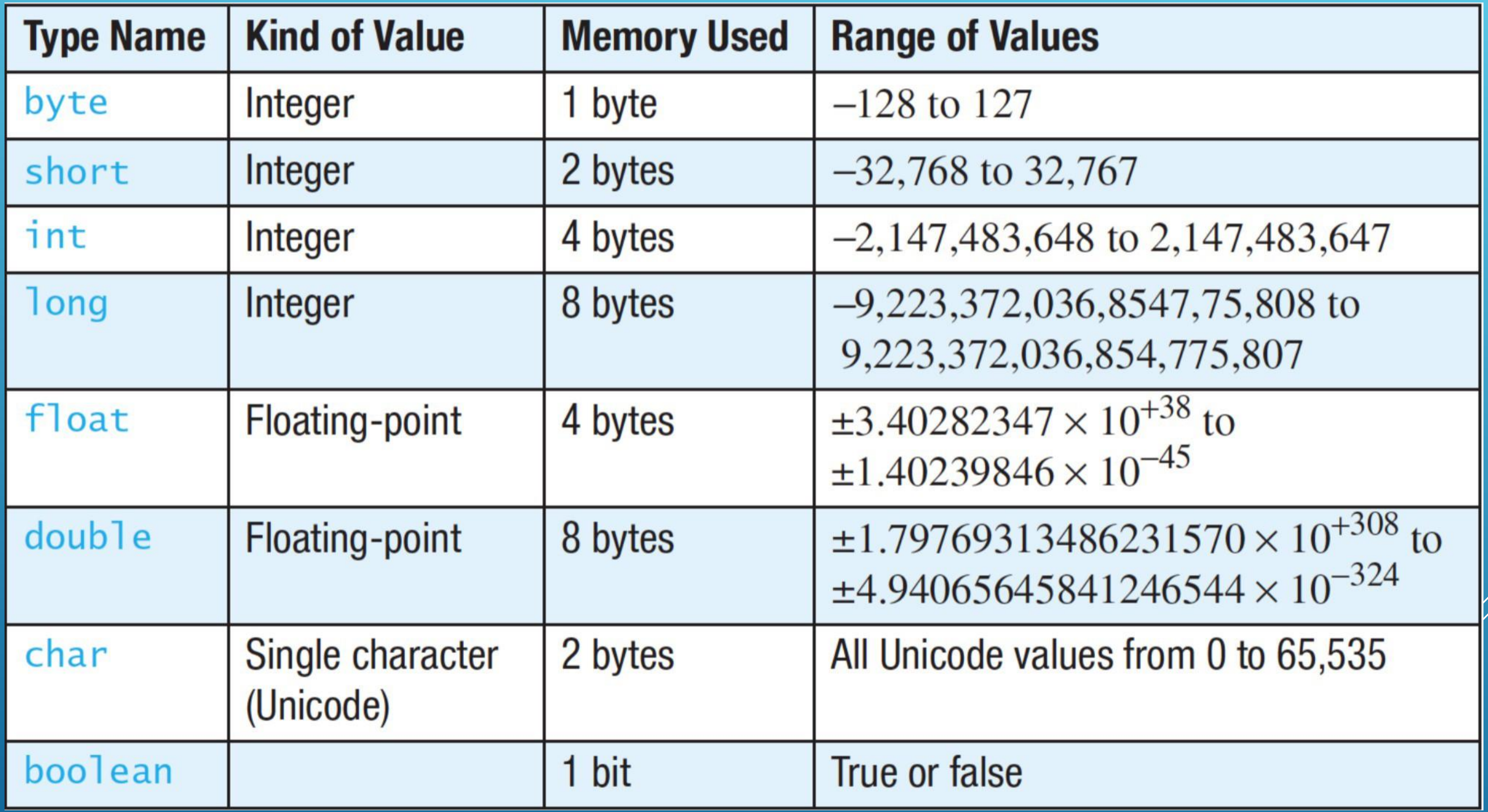
HOW TO DECLARE AND ASSIGN A VARIABLE WITH A VALUE?



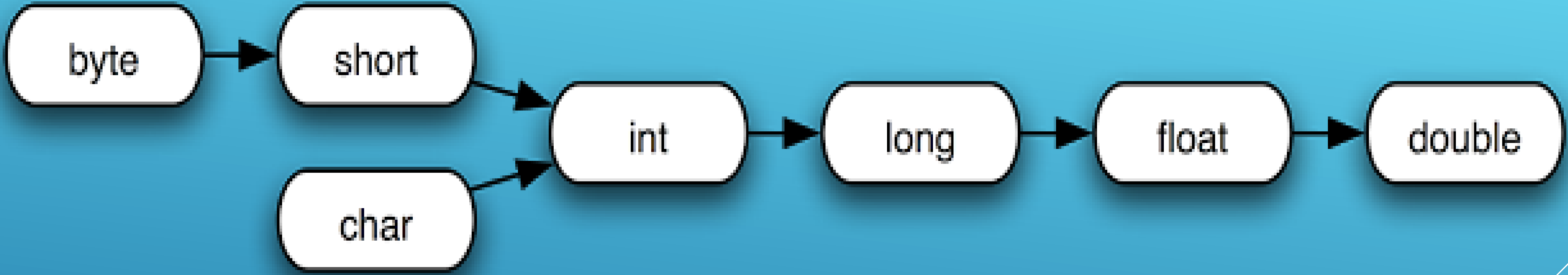
```
int quantity = 3;
```

DATA TYPES IN JAVA



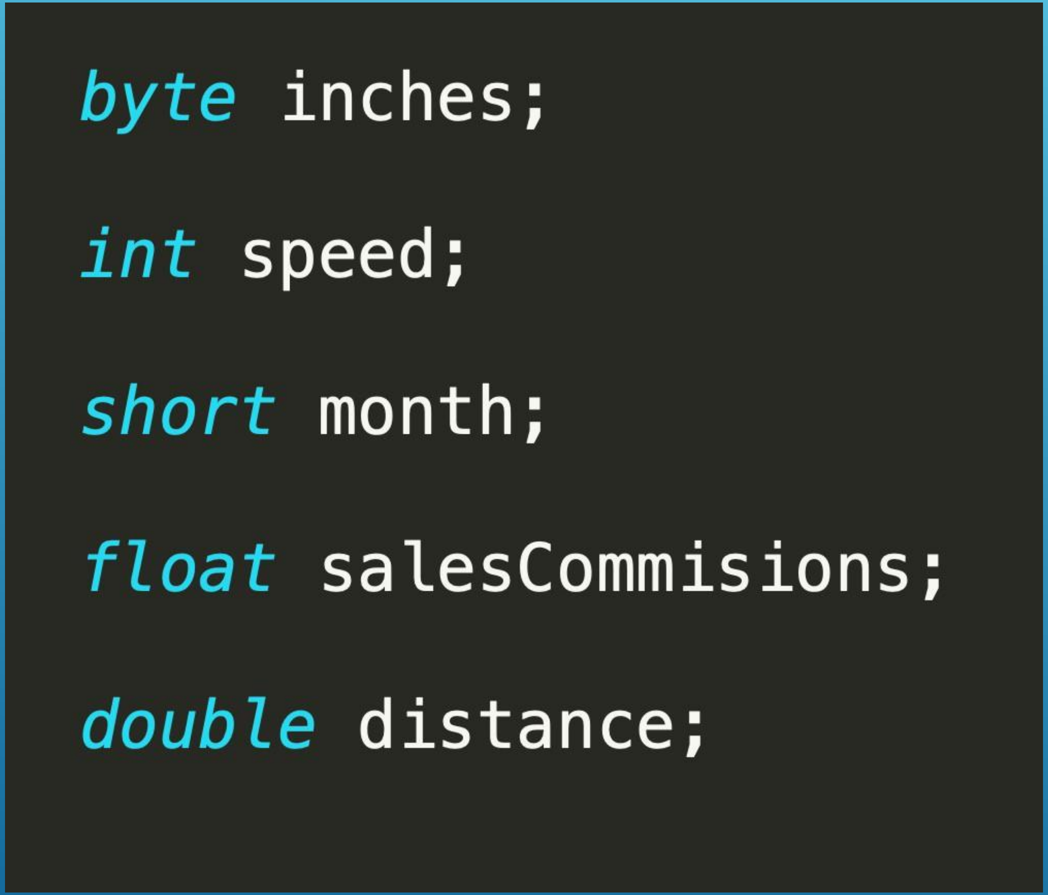


Type Name	Kind of Value	Memory Used	Range of Values
`byte`	Integer	1 byte	–128 to 127
`short`	Integer	2 bytes	–32,768 to 32,767
`int`	Integer	4 bytes	–2,147,483,648 to 2,147,483,647
`long`	Integer	8 bytes	–9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807
`float`	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
`double`	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
`char`	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
`boolean`		1 bit	True or false



boolean, char, byte, short, int, long, float, double

EXAMPLE OF VARIABLE DECLARATIONS



```
byte inches;

int speed;

short month;

float salesCommisions;

double distance;
```

THE INTEGER DATA TYPES

- BYTE, SHORT, INT, AND LONG ARE ALL INTEGER DATA TYPES.
- AN INTEGER VARIABLE CAN HOLD WHOLE NUMBERS SUCH AS 7, 125, -14, AND 6928.
- THE COMPILER WILL READ WHOLE NUMBERS AS INT DATATYPE BY DEFAULT

FLOATING-POINT DATA TYPES

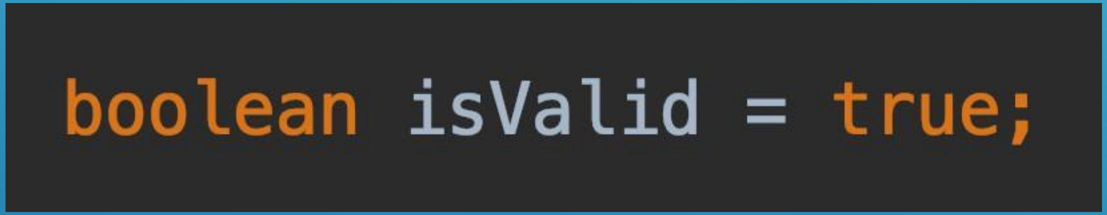
- FLOAT AND DOUBLE ARE FLOAT DATA TYPES.

- WHOLE NUMBERS ARE NOT ADEQUATE FOR MANY JOBS. IF YOU ARE WRITING A PROGRAM WORKS WITH DOLLAR AMOUNTS OR PRECISE MEASUREMENTS, YOU NEED A DATA TYPE THAT ALLOWS FRACTIONAL VALUES.
- VALUES SUCH AS 1.7 AND -45.316 ARE FLOATING-POINT NUMBERS.

- WHEN YOU WRITE A FLOATING-POINT LITERAL IN YOUR PROGRAM CODE, JAVA ASSUMES IT TO BE OF THE DOUBLE DATA TYPE.
- YOU CAN FORCE A DOUBLE LITERAL TO BE TREATED AS A FLOAT BY SUFFIXING IT WITH THE LETTER F OR F.

THE BOOLEAN DATA TYPE

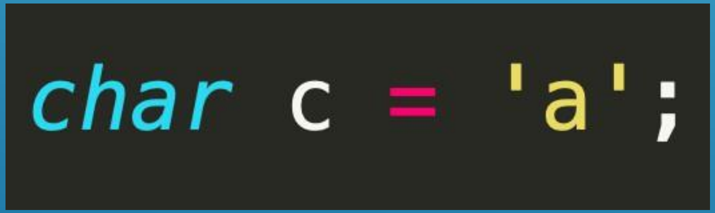
- THE BOOLEAN DATA TYPE ALLOWS YOU TO CREATE VARIABLES THAT MAY HOLD ONE OF TWO POSSIBLE VALUES: TRUE OR FALSE.



```
boolean isValid = true;
```

THE CHAR DATA TYPE

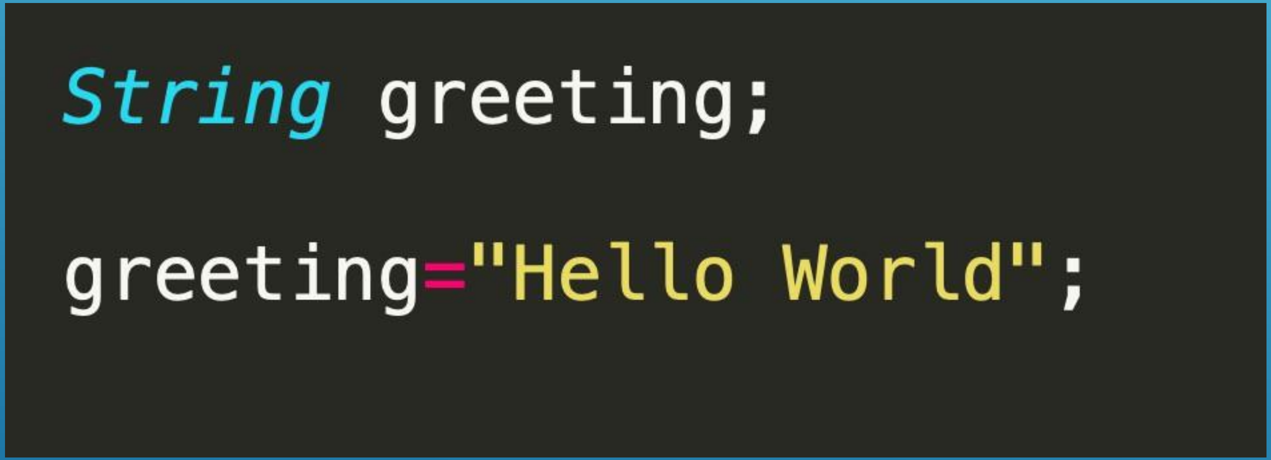
- THE CHAR DATA TYPE IS USED TO STORE CHARACTERS.
- A VARIABLE OF THE CHAR DATA TYPE CAN HOLD ONE CHARACTER AT A TIME.
- CHARACTER LITERALS ARE ENCLOSED IN SINGLE QUOTATION MARKS.



```
char c = 'a';
```

STRING

- STRING IS A SEQUENCE OF CHARACTERS, SURROUNDED BY DOUBLE QUOTES(“”)



```
String greeting;

greeting="Hello World";
```

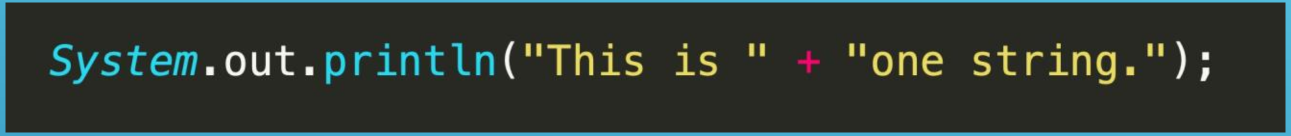
CONCATENATION

THE ACTION OF LINKING THINGS TOGETHER.

+

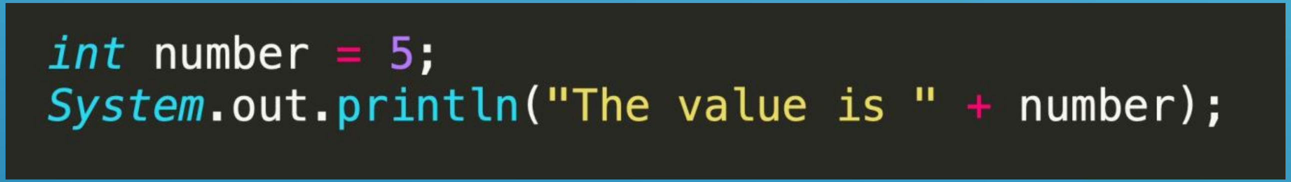
String Literal		String Variable		String Literal		Number Variable		String Literal
“Hi, my name is”	+	name	+	“ and I am “	+	years	+	“years old”

CONCATENATION



```java
System.out.println("This is " + "one string.");
```

This is one string.



```java
int number = 5;
System.out.println("The value is " + number);
```

The value is 5

Note: When you concatenate anything to a String, it will become a String

UNICODE

- CHARACTERS ARE INTERNALLY REPRESENTED BY NUMBERS.
- EACH PRINTABLE CHARACTER, AS WELL AS MANY NON-PRINTABLE CHARACTERS, IS ASSIGNED A UNIQUE NUMBER.
- JAVA USES UNICODE, WHICH IS A SET OF NUMBERS THAT ARE STORED AS CODES FOR REPRESENTING CHARACTERS.
- WHEN A CHARACTER IS STORED IN MEMORY, IT IS ACTUALLY THE NUMERIC CODE THAT IS STORED. WHEN THE COMPUTER IS INSTRUCTED TO PRINT THE VALUE ON THE SCREEN, IT DISPLAYS THE CHARACTER THAT CORRESPONDS WITH THE NUMERIC CODE.
- HTTPS://UNICODE-TABLE.COM/EN/

CREATING VARIABLES

1

```
int price=5;
int quantity=14;
int total=price*quantity;
```

2

```
int price,quantity,total;
price=5;
quantity=14;
total=price*quantity;
```

3

```
int price=5,quantity=14;
int total=price*quantity;
```

DECLARING MULTIPLE VARIABLES

```java
String s1,s2;
String s3 = "yes", s4 = "no";
```

```java
int i1,i2,i3=0;
```

```java
int num,String value; //Does not compile
```

TASK – 1

CREATE 3 VARIABLES AND ASSIGN DIFFERENT VALUES DISPLAY EACH OF THEM IN THE CONSOLE.

TASK – 2

WRITE A JAVA PROGRAM TO DECLARE ONE INTEGER VARIABLES, ONE FLOAT VARIABLE, AND ONE STRING VARIABLE AND ASSIGN 10, 12.5, AND "JAVA PROGRAMMING" TO THEM RESPECTIVELY. THEN DISPLAY THEIR VALUES ON THE SCREEN.

TASK – 4

STORE THE FOLLOWING INTO VARIABLES: NUMBER OF CHILDREN, PARTNER'S NAME, GEOGRAPHIC LOCATION, JOB TITLE.
OUTPUT YOUR FORTUNE TO THE SCREEN LIKE SO: "YOU WILL BE A X IN Y, AND MARRIED TO Z WITH N KIDS."

X – JOB TITLE
Y – LOCATION
Z – PARTNER'S NAME
N - # OF KIDS

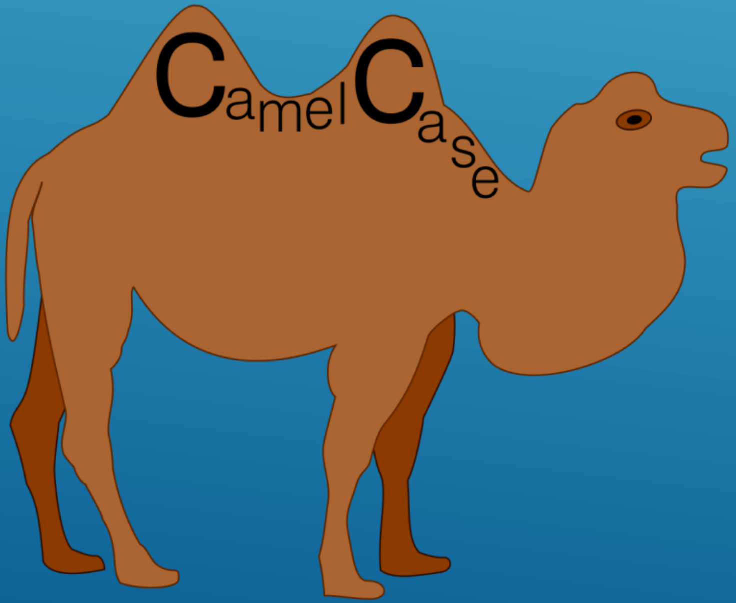
TASK – 3

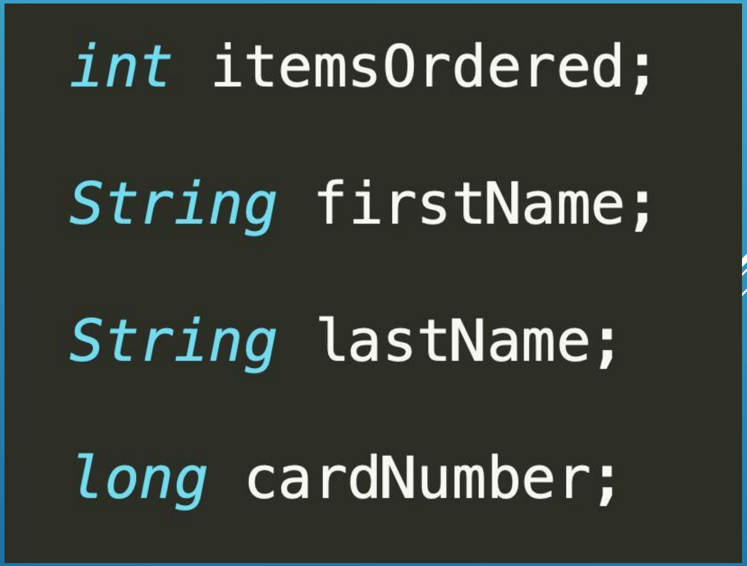
WRITE A JAVA PROGRAM CODE FOR:

ASSIGN THE VALUE "IN 1491 COLUMBUS SAILED THE OCEANBLUE" TO AN APPROPRIATE VARIABLE,WRITE A PROGRAM IN JAVA TO CHANGE THE YEAR IN THE STATEMENT ABOVE FROM 1491 TO 1492.

RULES FOR IDENTIFIERS(VARIABLE OR CLASS NAMES)

- VARIABLE NAMES SHOULD BE READABLE.
- WHEN VARIABLE NAMES CONTAIN TWO OR MORE WORDS, USE CAMEL CASE.



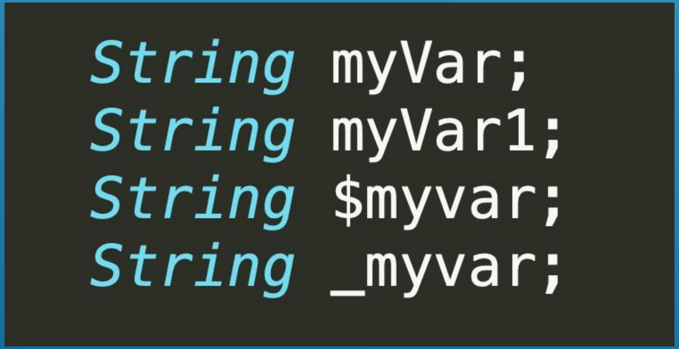


```
int itemsOrdered;

String firstName;

String lastName;

long cardNumber;
```

RULES FOR IDENTIFIERS(VARIABLE OR CLASS NAMES)

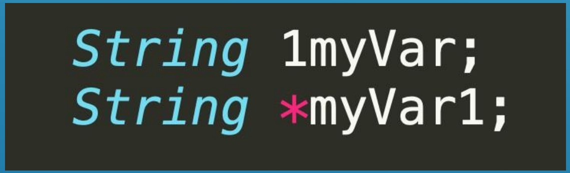
- THE FIRST CHARACTER MUST BE ONE OF THE LETTERS A-Z OR A-Z, AN UNDERSCORE(_), OR A DOLLAR SIGN($)

Valid Example



```
String myVar;
String myVar1;
String $myvar;
String _myvar;
```

Invalid Example



```
String 1myVar;
String *myVar1;
```

RULES FOR IDENTIFIERS(VARIABLE OR CLASS NAMES)

- AFTER THE FIRST CHARACTER, YOU MAY USE THE LETTERS A-Z OR A-Z, THE DIGITS 0-9, UNDERSCORES(_), OR DOLLAR SIGN($)
- IDENTIFIERS CAN NOT INCLUDE SPACES

Valid Example

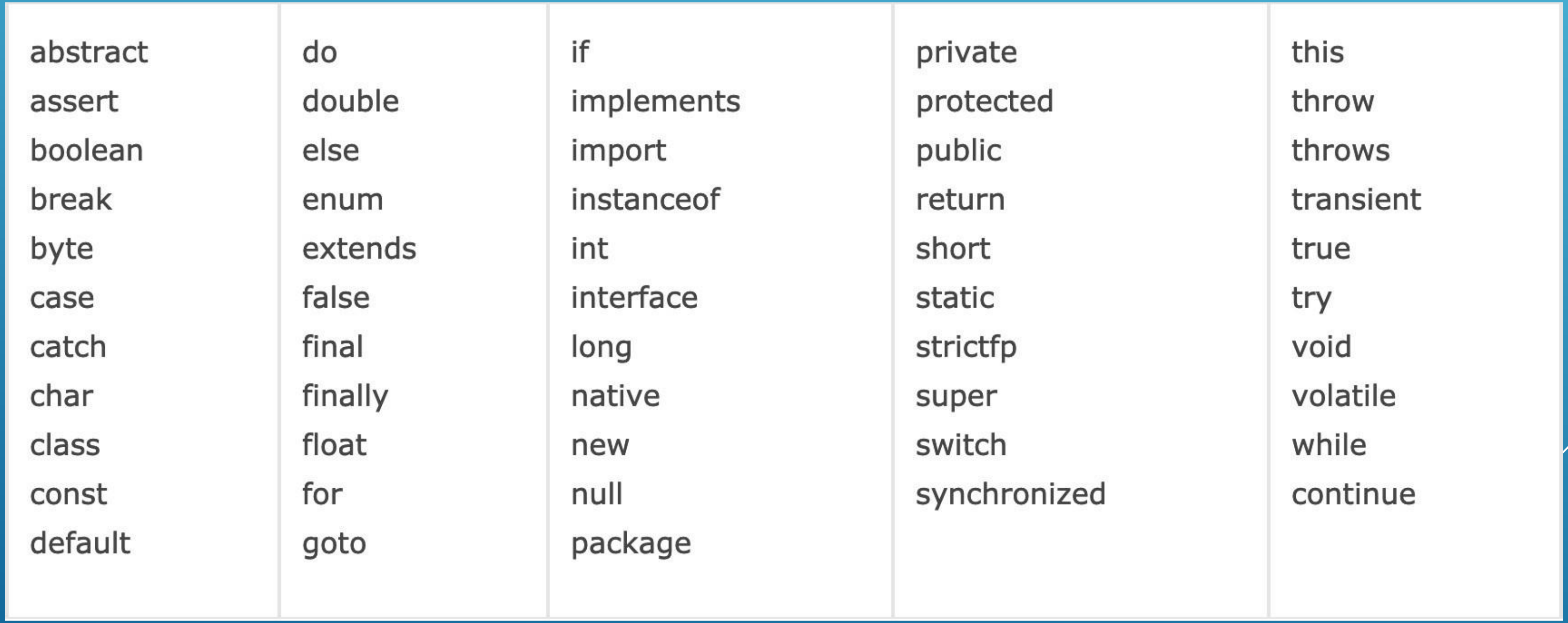
```
String my20thBirthday$;
int ageOfThe_Employer100;
```

Invalid Example

```
int ageOfThe@Employer100;
String my20th Birthday$;
```

RULES FOR IDENTIFIERS(VARIABLE OR CLASS NAMES)

- VARIABLE NAMES CAN NOT BE JAVA RESERVED WORD



abstract	do	if	private	this
assert	double	implements	protected	throw
boolean	else	import	public	throws
break	enum	instanceof	return	transient
byte	extends	int	short	true
case	false	interface	static	try
catch	final	long	strictfp	void
char	finally	native	super	volatile
class	float	new	switch	while
const	for	null	synchronized	continue
default	goto	package		

4THEDOGS	🚫	value2B	✓
MY NAME	🚫	lastval!	🚫
MY-NAME	🚫	ba#!@t	🚫
LAST_NAME	✓	$12variable	✓

TASK – 5

WHICH OF THE FOLLOWING ARE VALID JAVA IDENTIFIERS?

DAYOFWEEK	A$B
3DGRAPH	_HELLOWORLD
JUNE1997	TRUE
MIXTURE#3	PUBLIC
WEEK DAY	1980_S

TASK - 6

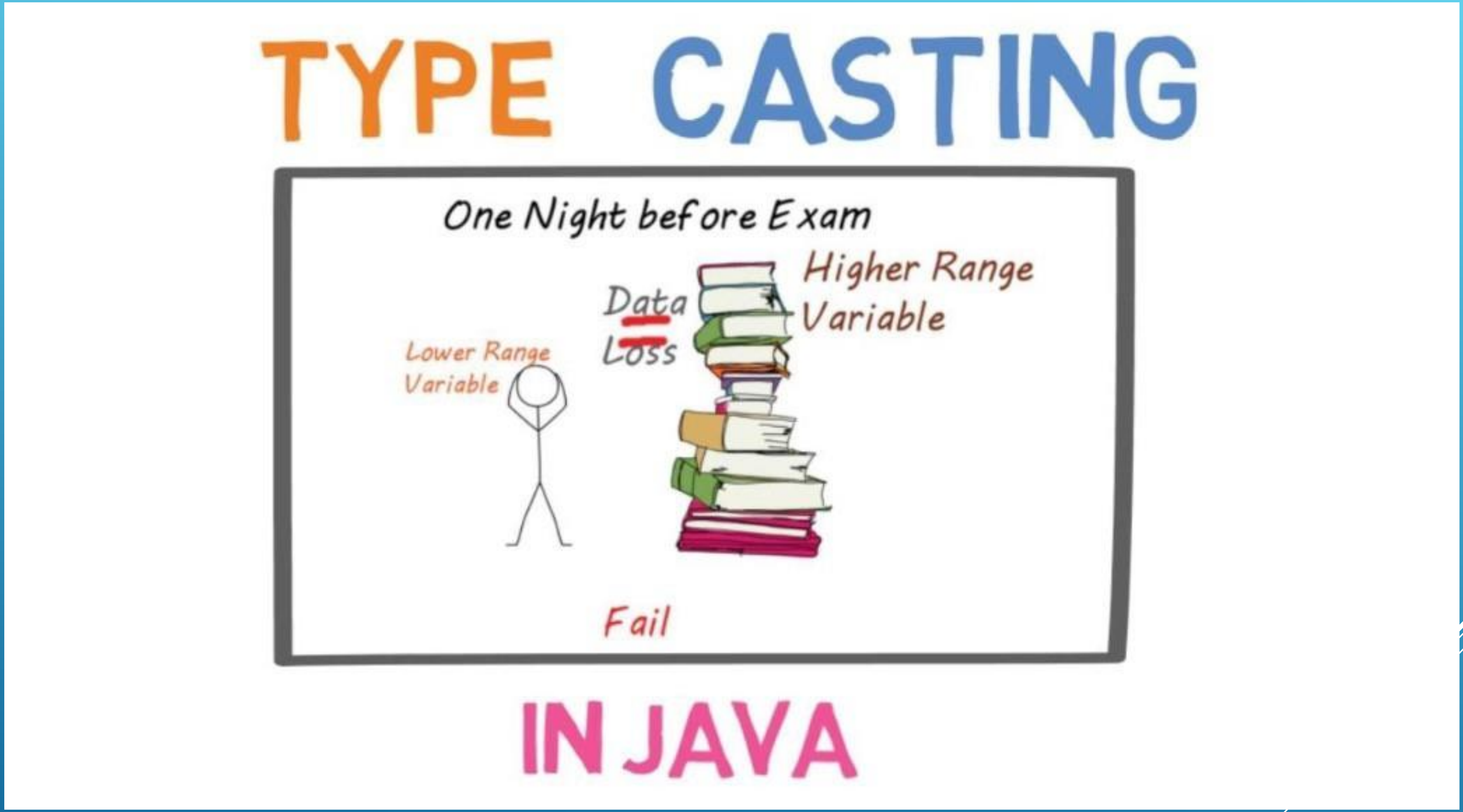
DECLARE A VARIABLE WHOSE NAME IS THE COMBINATION OF 3 CHARACTERS, IN THIS ORDER: A LEGAL FIRST CHARACTER THAT IS NOT A LETTER, A CHARACTER THAT WOULD BE ILLEGAL AS A FIRST CHARACTER, AND A CHARACTER THAT WOULD BE LEGAL ANYWHERE IN THE NAME

TASK - 7

1 DECLARE A VARIABLE WHOSE NAME IS YOUR FIRST AND LAST NAMES COMBINED, IN CAMELCASE

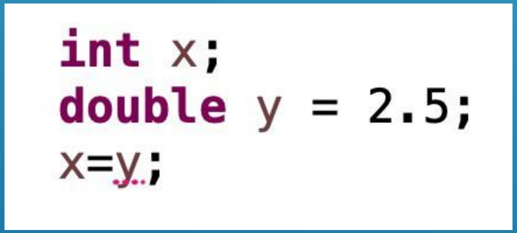
2 ASSIGN THE VARIABLE YOUR FIRST AND LAST NAMES, AS A STRING

3 PRINT IN THE CONSOLE, SPECIFYING THE VARIABLE, NOT THE STRING, AS THE MESSAGE

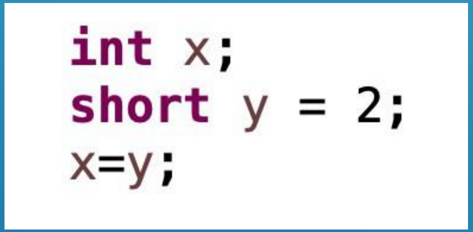

TYPE CASTING
One Night before Exam
Higher Range Variable
Data Loss
Lower Range Variable
Fail
IN JAVA

CONVERSION BETWEEN PRIMITIVE DATA TYPES

BEFORE A VALUE CAN BE STORED IN A VARIABLE, THE VALUE'S DATA TYPE MUST BE COMPATIBLE WITH THE VARIABLE'S DATA TYPE. JAVA PERFORMS SOME CONVERSIONS BETWEEN DATA TYPES AUTOMATICALLY BUT DOES NOT AUTOMATICALLY PERFORM ANY CONVERSION THAT CAN RESULT IN THE LOSS OF DATA.

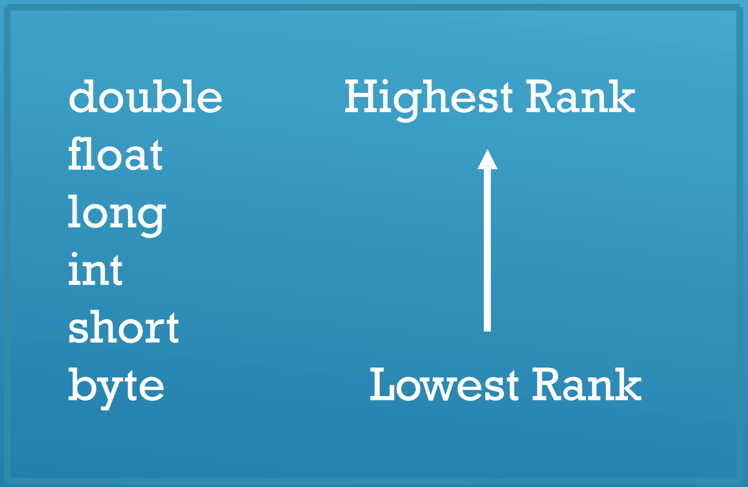


```
int x;
double y = 2.5;
x=y;
```

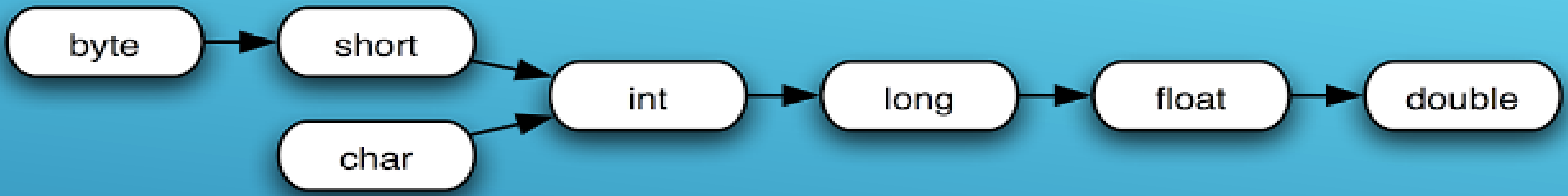


```
int x;
short y = 2;
x=y;
```

WHY DOES JAVA PERMIT A SHORT TO BE STORED IN AN INT, BUT DOES NOT PERMIT A DOUBLE TO BE STORED IN AN INT?



ALL DATATYPES CAN BE CAST IMPLICITLY FROM A SMALLER SIZE TO A BIGGER SIZE



Something like this happens implicitly:

```
int one = 1;
long num = one;
```

Similar scenario as above but flipped:

```
long num = 1;
int one = num; This is incorrect
int one = (int) num; This is correctly cast
```

** BUT if you want to convert a datatype from a bigger size to a smaller sizer you must do it explicitly

WIDENING CONVERSION

IN STATEMENTS WHERE VALUES OF LOWER-RANKED DATA TYPES ARE STORED IN VARIABLES OF HIGHER-RANKED DATA TYPES, JAVA AUTOMATICALLY CONVERTS THE LOWER-RANKED VALUE TO THE HIGHER-RANKED TYPE. THIS IS CALLED WIDENING-CONVERSION.

```
double x;
int y=10;
x=y;        //Performs a widening conversion
```

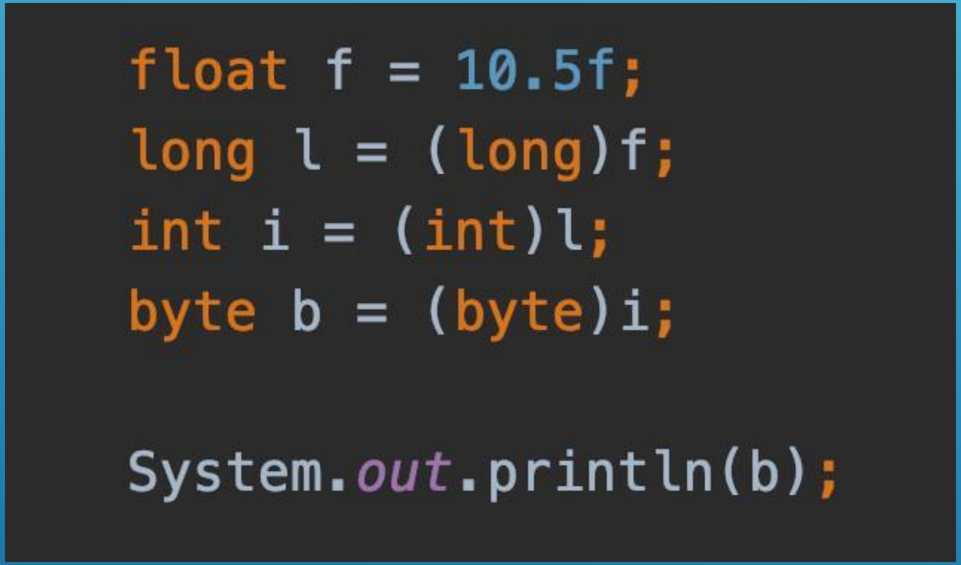
NARROWING CONVERSION

A NARROWING CONVERSION IS THE CONVERSION OF A VALUE TO A LOWER-RANKED TYPE. FOR EXAMPLE CONVERTING A DOUBLE TO AN INT WOULD BE A NARROWING CONVERSION. BECAUSE NARROWING CONVERSION CAN POTENTIALLY CAUSE A LOSS OF DATA, JAVA DOES NOT AUTOMATICALLY PERFORM THEM.

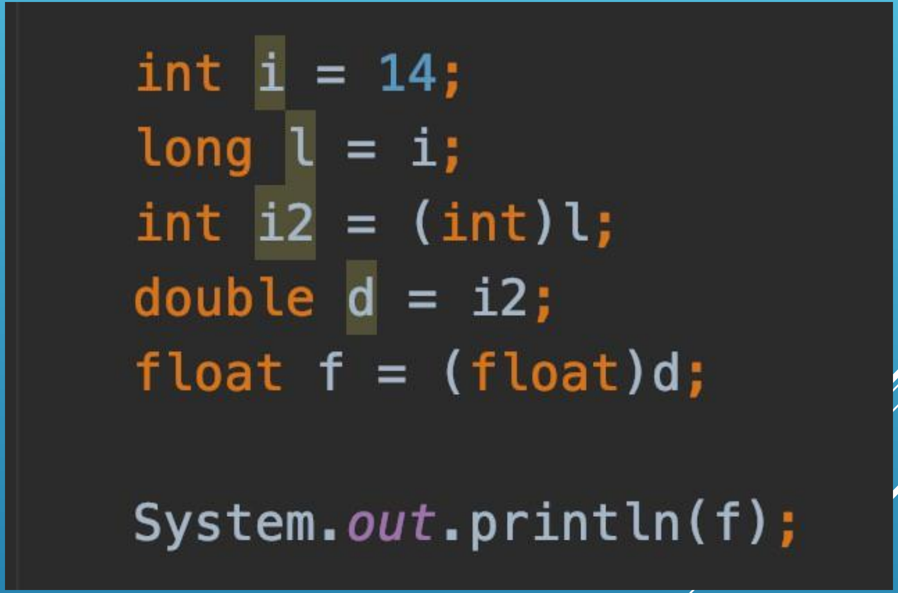
Casting Formula:

smallerDatatype varName = (smallerDatatype) biggerDatatypeValue

CASTING EXAMPLES



```
float f = 10.5f;
long l = (long)f;
int i = (int)l;
byte b = (byte)i;

System.out.println(b);
```



```
int i = 14;
long l = i;
int i2 = (int)l;
double d = i2;
float f = (float)d;

System.out.println(f);
```