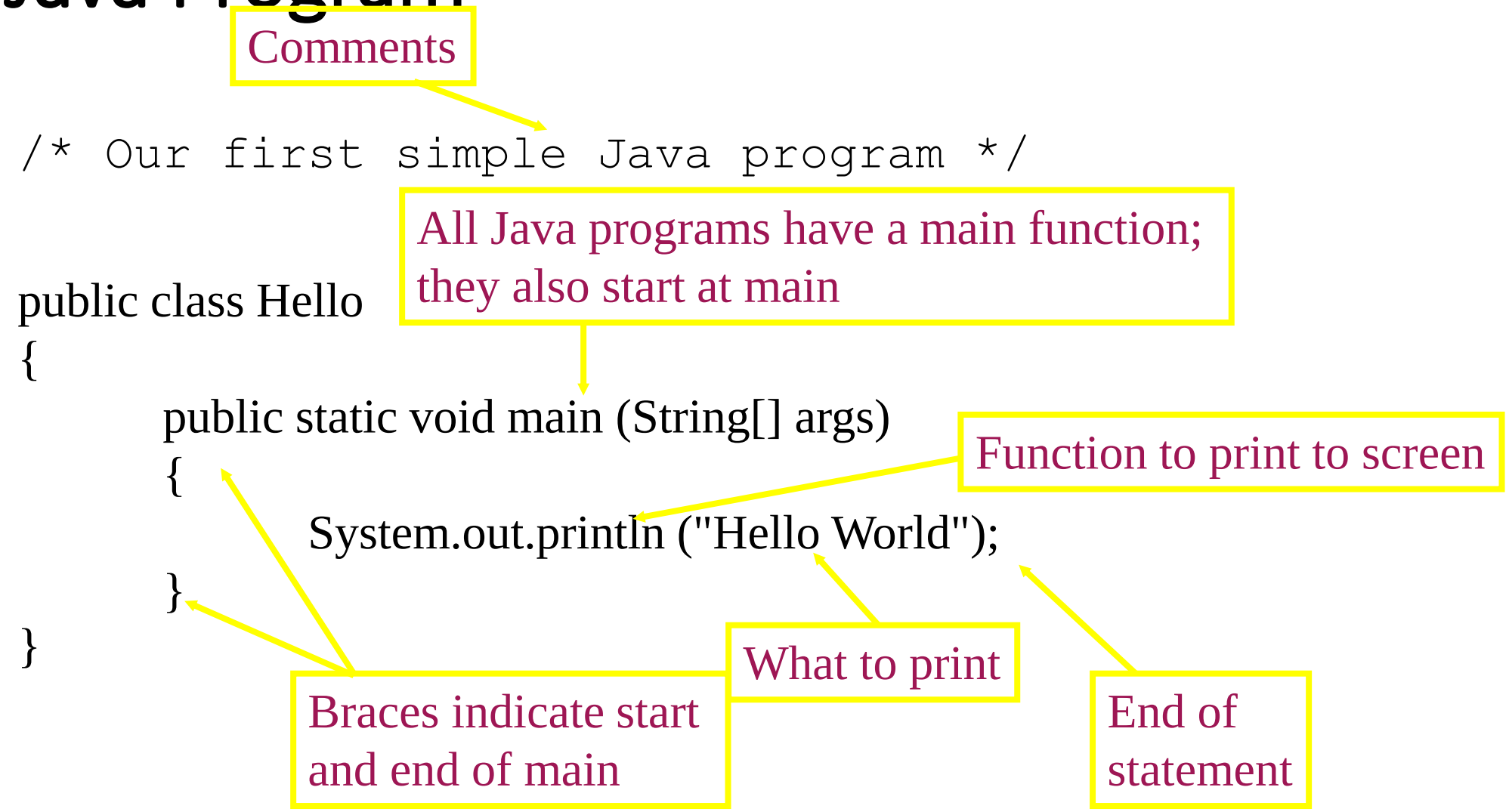
Lecture: 2
OOP Basic

Sub: Object Oriented Programming

Code: CSE-1205

Google Classroom Code: 7xtirlv

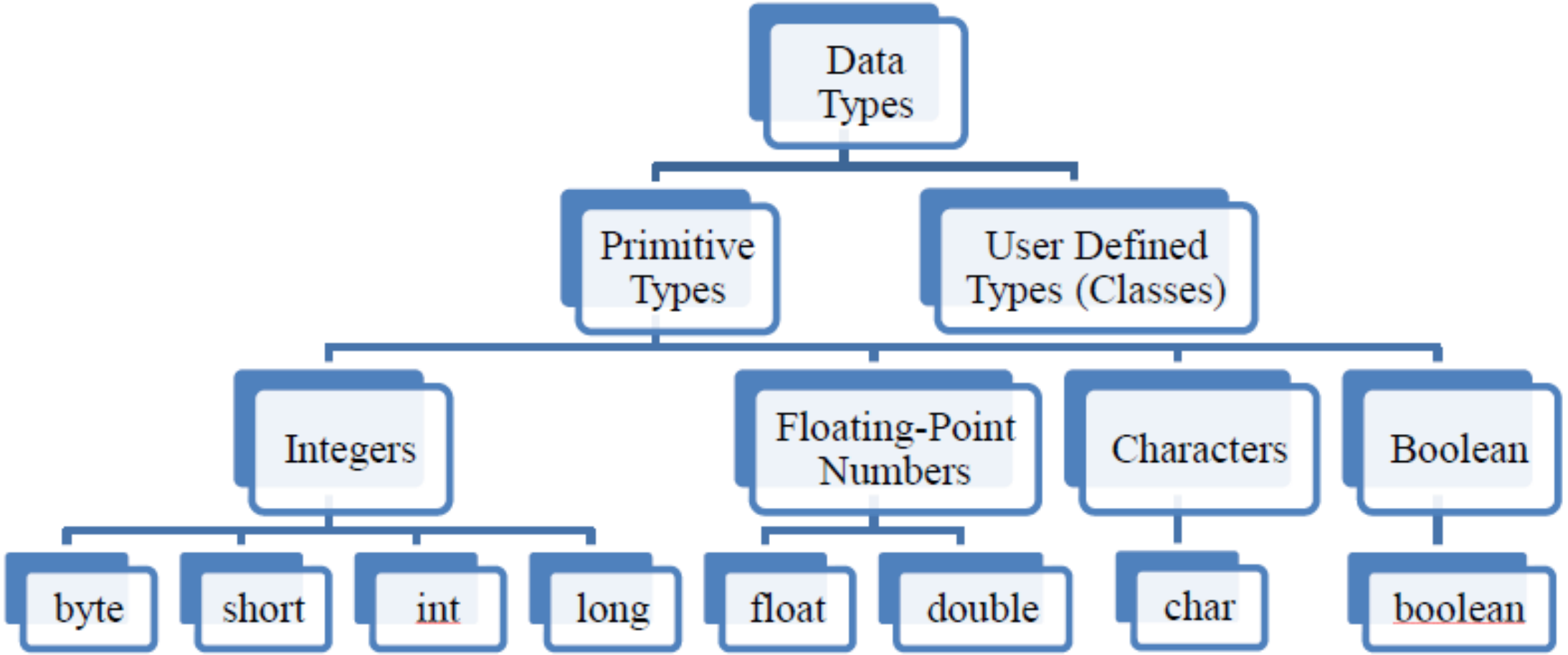
First Java Program



Identifiers and Keywords

- Identifiers are names for variables, classes, methods etc.
- Good ones are compact, but inidicate what they stand for
  - radius, width, height, length
- Java is case sensitive (so as identifier).
- **Rules**:
  - May contain upper case, lower case letters, numbers, underscore, dollar sign.
  - Must not begin with a number.
- **Keyword:**
  - Some words are reserved, and can't be used as identifiers

Java Data Types



Java Data Types

- There are 8 primitive types in Java.
- **Integer type:**

byte	An 8-bit signed integer.
short	A 16-bit signed integer.
int	A 32-bit signed integer.
long	A 64-bit signed integer.

Java Data Types

- **Floating point types:**

Float	A 32-bit IEEE floating point.
double	A 64-bit IEEE floating point.

- **Other types:**

boolean	Either true or false.
char	A 16-bit Unicode character.

Primitive Variable Types

- Java has 8 (or so) primitive types:
  - float
  - double
  - boolean
  - char
  - byte
  - short
  - int
  - long

Real number

Two Values: true and false

Integer Number

Primitive real (floating-point) types

- A float takes up 4 bytes of space
  - Has **6** decimal places of accuracy: 3.14159

- A double takes up 8 bytes of space
  - Has **15** decimal places of accuracy: 3.14159265358979

- **Always use doubles**
  - It will save you quite a headache!

Primitive integer types

- Consider a byte:

- A Java byte can have values from -128 to 127
  - From -2^7 to 2^7-1
- C/C++ has **unsigned versions; Java does not**
- What would be the result for the following program?

```
byte a = 127;
a+=1;
System.out.println(a);
```

The Result will be: ? -128

Primitive Character Type

- All characters have a integer equivalent
  - ‘0’ = 48
  - ‘1’ = 49
  - ‘A’ = 65
  - ‘a’ = 97

- Thus, you can refer to ‘B’ as ‘A’+1

- Example:
  - char var=‘a’; or, char var=97;
  - var++; //now, var=‘b’
- There are **no negative char**. So the range of char is 0-65536

Primitive boolean type

- The boolean type has only two values:
  - true
  - false
- Example:
  - boolean var=true;
- There are boolean-specific operators
  - **&&** is and
  - || is or
  - **!** is not
  - etc.

Variable initialization

- Consider the following code:

```
int x;
System.out.println(x);
```

- What happens?

- **Error message:**
  - variable x might not have been initialized

- **Java requires you to give x a value before you use it**

Printing variables

- To print a variable to the screen, put it in a System.out.println() statement:

    - int x = 5;
    - System.out.println (“The value of x is ” + x);
    - System.out.print (x);

- Important points:
    - **Strings are enclosed in double quotes**
    - If there are **multiple parts** to be printed, they are separated by a **plus sign**

Constants

- Consider the following:

```
final int x = 5;
```

- The value of x can **NEVER be changed**!
  - The value assigned to it is “final”

- This is how Java defines constants

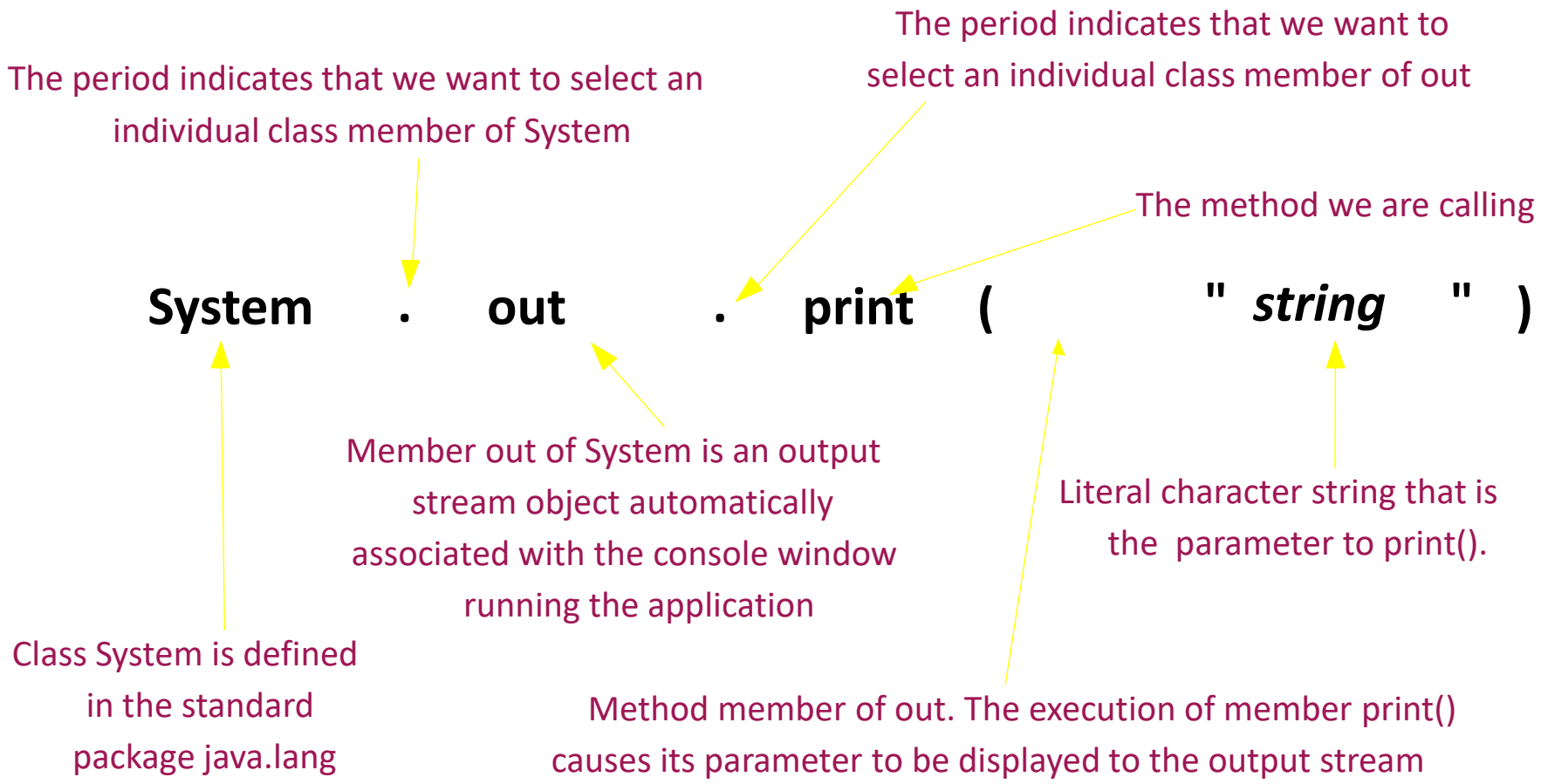
Take Input and Print output

- **System.out**
  - Prints to standard output
  - Equivalent to “cout” in C++, and “printf()” in C

- **System.in**
  - Reads from standard input
  - Equivalent to “cin” in C++, and “scanf()” in C

- **System.err**
  - Prints to standard error
  - Equivalent to “cerr” in C++, and “fprintf(stderr)” in C

Print Output



Taking input from the keyboard

- Here **Scanner class** is used to take input from the keyboard.
- Scanner is a simple **text scanner** which can **parse primitive types and strings** using regular expressions.
- First, Scanner class is **connected to System.in**
- Then, it uses it's internal functions to read from System.in
- Scanner class is under the package of **java.lang.util**
    - Example:

```
Scanner user_input = new Scanner( System.in );
String first_name;
System.out.print("Enter your first name: ");
first_name = user_input.next( );
```

Example

1.

```
Scanner sc = new Scanner(System.in);
    int i;
    If(sc.hasNextInt()==true)
          i = sc.nextInt();
       else{
       }
```

2.

```
import java.util.*;
public static void main(String[] args) {
   double value;
   System.out.print("Enter a floating point number:");

   Scanner stdin = new Scanner(System.in);
   if(stdin.hasNextDouble()==true)
              value=stdin.nextDouble();
   System.out.println("You have entered: "+value);
}
```

Scanner API

1. public Scanner(InputStream in) // Scanner(): convenience constructor for an
   // InputStream
1. public Scanner(File s) // Scanner(): convenience constructor for a filename

2. public int nextInt() // nextInt(): next input value as an int

3. public short nextShort() // nextShort(): next input value as a short

4. public long nextLong() // nextLong(): next input value as a long

5. public double nextDouble() // nextDouble(): next next input value as a double

6. public float nextFloat() // nextFloat(): next next input value as a float

7. public String next() // next(): get next whitespace-free string

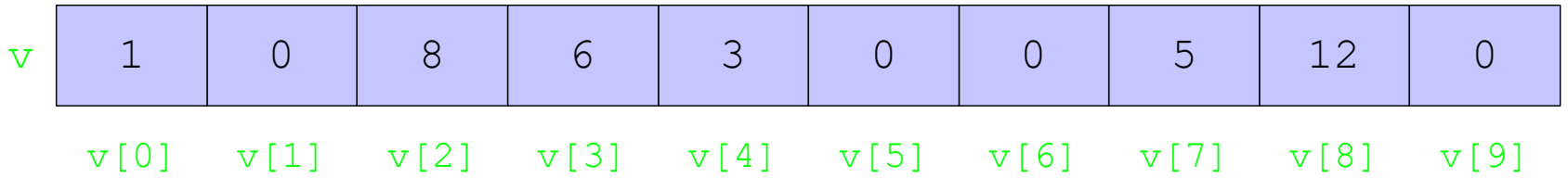
8. public String nextLine() // nextLine(): return contents of input line buffer

9. public boolean hasNext() // hasNext(): is there a value to next

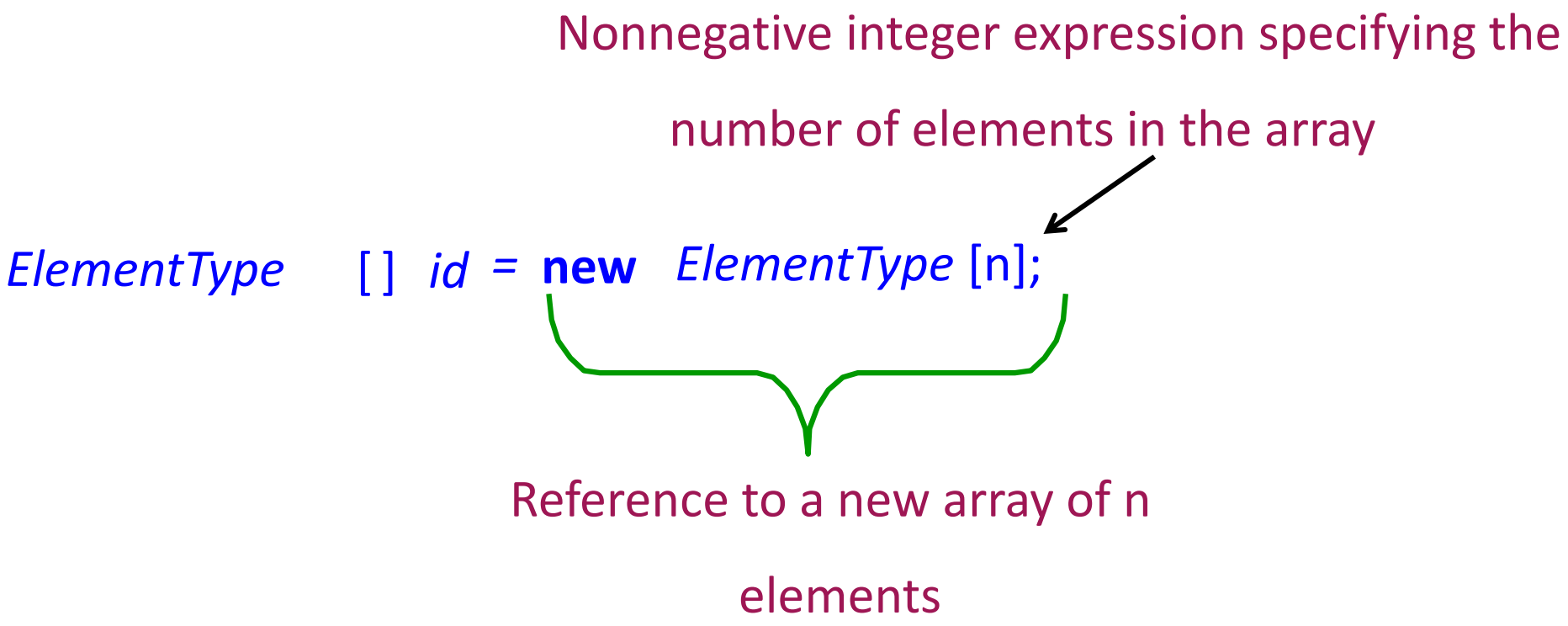
Array

- Programmer often need the ability to represent a group of values as a list
  - List may be one-dimensional or multidimensional

- Java provides arrays and the collection classes
  - The Vector class is an example of a collection class



Array Syntax



Array Example

- Example

```
String[] puppy = { “pika“, “arlo“, “schuyler", “nikki“ };
int[] unit = { 1 };
```

- Equivalent to

```
String[] puppy = new String[4];
puppy[0] = “pika";          puppy[1] = “arlo";
puppy[2] = “schuyler";      puppy[3] = “nikki";

int[] unit = new int[1];
unit[0] = 1;
```

Where we've seen arrays

- public static void main (String[] args)
  - Thus, the main() method takes in a String array as the parameter

- Note that you can also define it as:
- public static void main (String args[])

String

- Java provides a class definition for a type called String
- Since the String class is part of the java.lang package, no special imports are required to use it (like a header file in C).
- Just like regular datatypes (and like C), variables of type String are declared as:
  - String s1;
  - String s2, s3; //etc.
- Note that String is uppercase. This is the Java convention for classnames.

String

- Initializing a String is painless
  - s1 = “This is some java String”;
- Note that double quotes are required.
- Memory is allocated dynamically.
- Think of above method as shortcut for more standard way (assuming s1 has been declared):
  - s1 = new String(“This is some java String”);
  - *new* operator required to create memory for new String object.

String Example

- Best to see by way of example:

```
String s = new String(“Hello”);
Char c = s.charAt(3);
System.out.println(c);
```

- Method charAt called on String object s taking single integer parameter.