Comp 401
Overview of Java
Conventional Programming

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

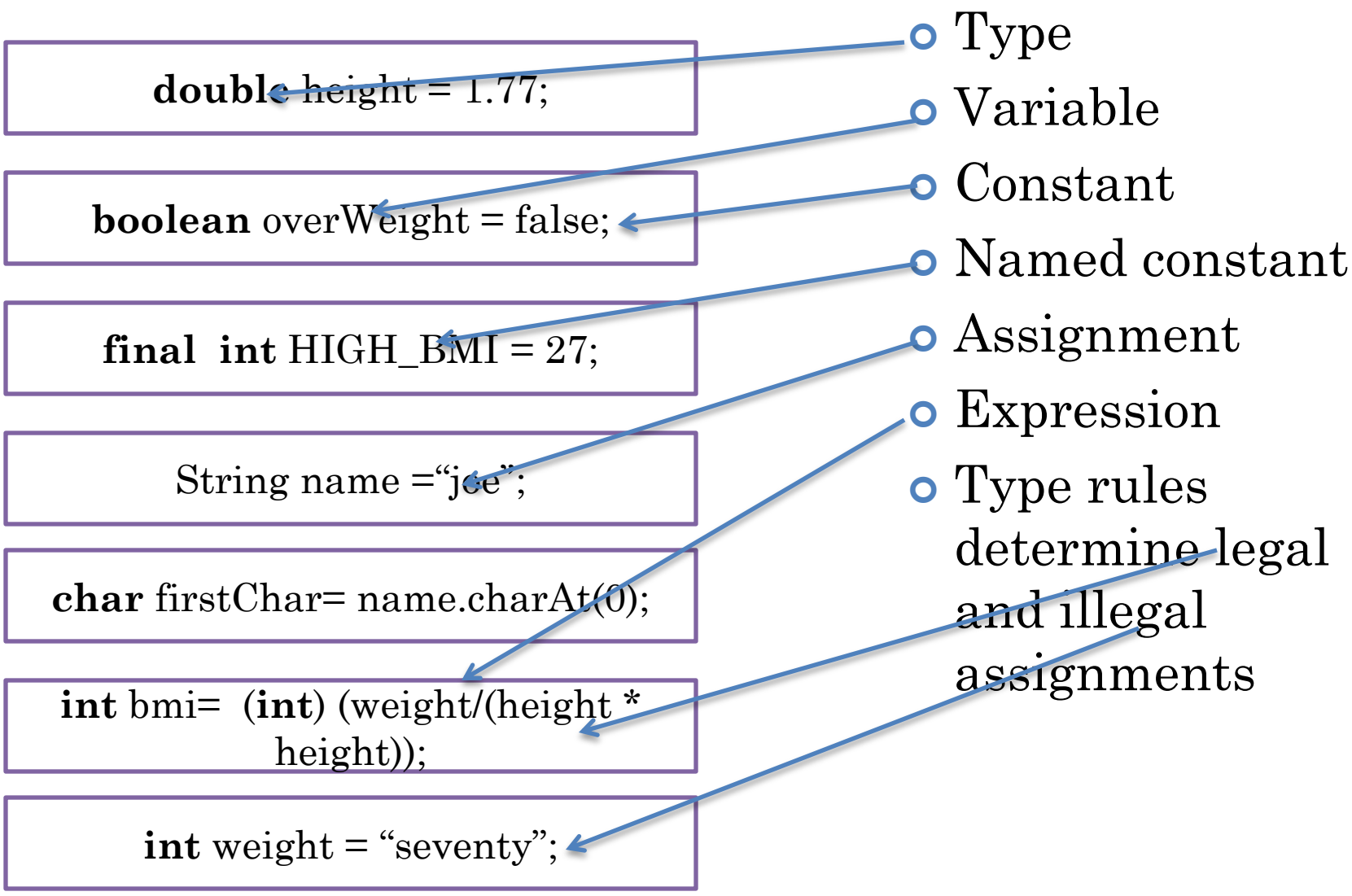
Brief Review

Help you recall the concepts

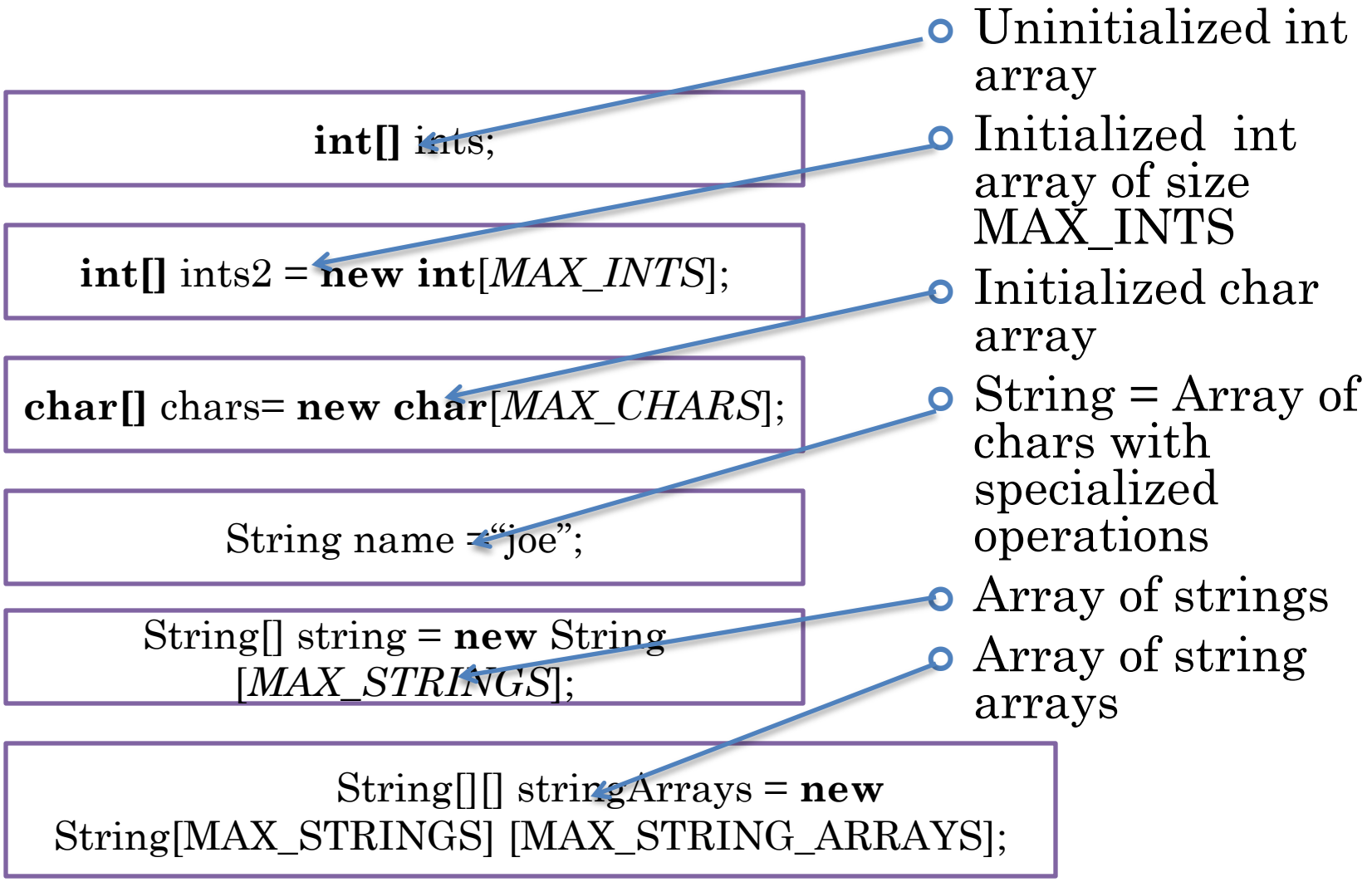
Show Java syntax to non Java programmers

Show what Java features you can use in first assignment

Types, Variables, Assignment, Constant, Expressions



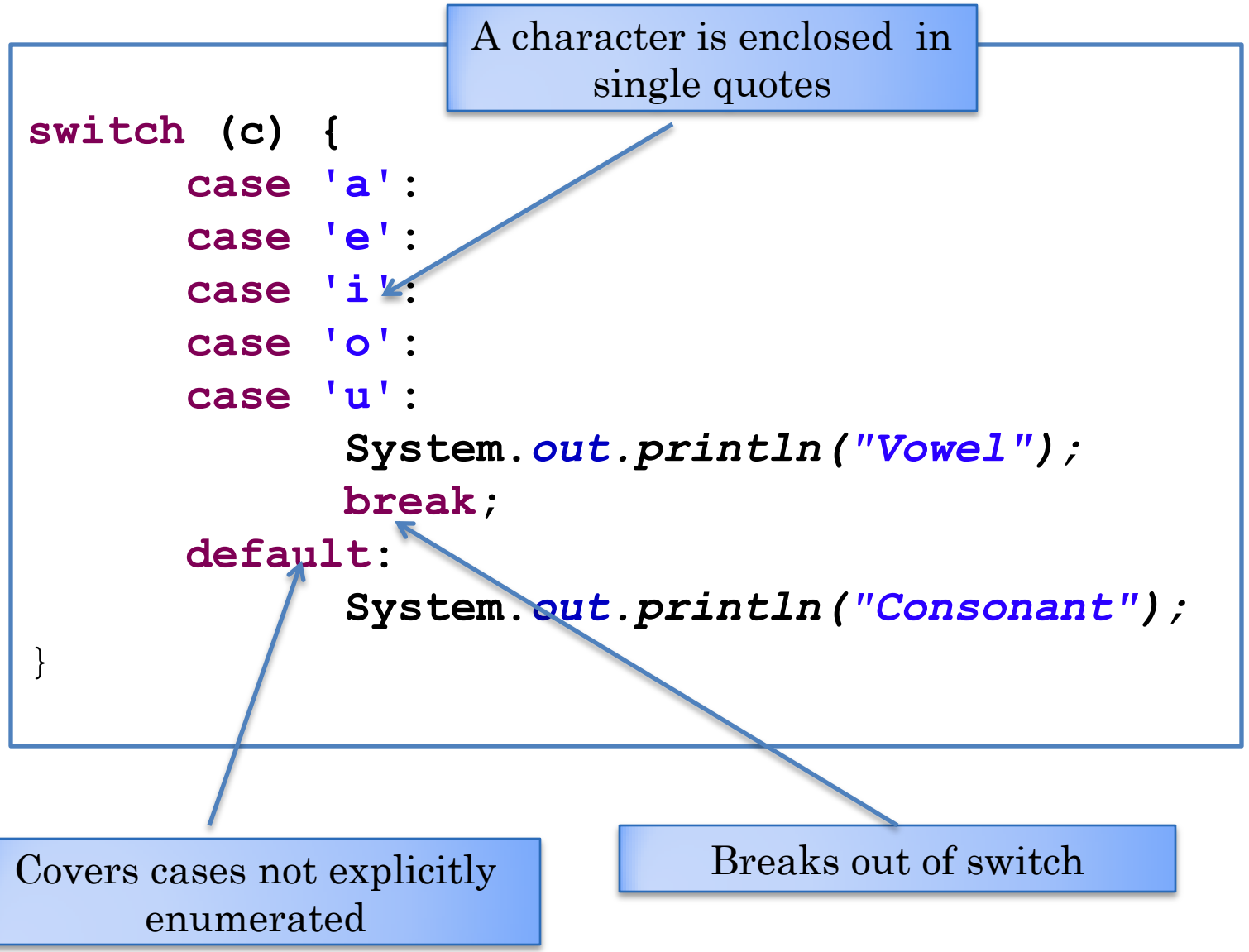
Arrays


int[] ints;
int[] ints2 = new int[MAX_INTS];
char[] chars= new char[MAX_CHARS];
String name =“joe”;
String[] string = new String [MAX_STRINGS];
String[][] stringArrays = new String[MAX_STRINGS] [MAX_STRING_ARRAYS];
Uninitialized int array
Initialized int array of size MAX_INTS
Initialized char array
String = Array of chars with specialized operations
Array of strings
Array of string arrays

Conditionals and Output

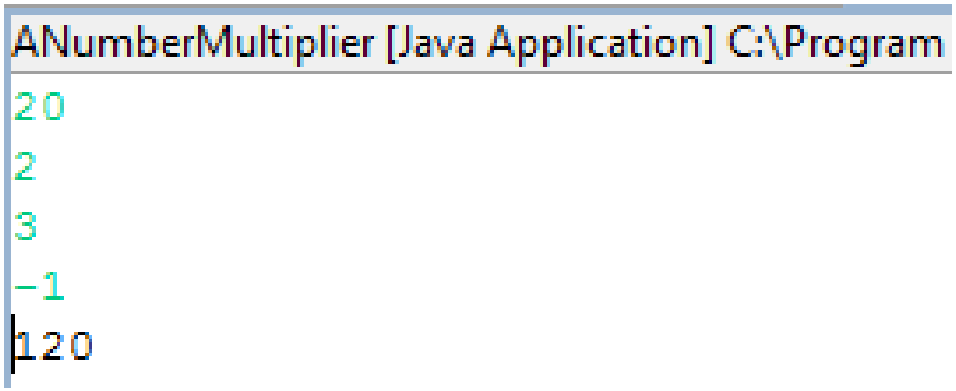
```
if (score < PASS_CUTOFF) {
        System.out.println("***************");
        System.out.println("FAILED");
        System.out.println("***************");
    }
else     {
        System.out.println("***************");
        System.out.println("PASSED");
        System.out.println("Congratulations!");
        System.out.println("***************");
   }
```

SWITCH: MULTIPLE CHOICE CONDITIONAL



While Loops

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product = product* nextNum;
        nextNum = Console.readInt();
}
System.out.print (product);
```



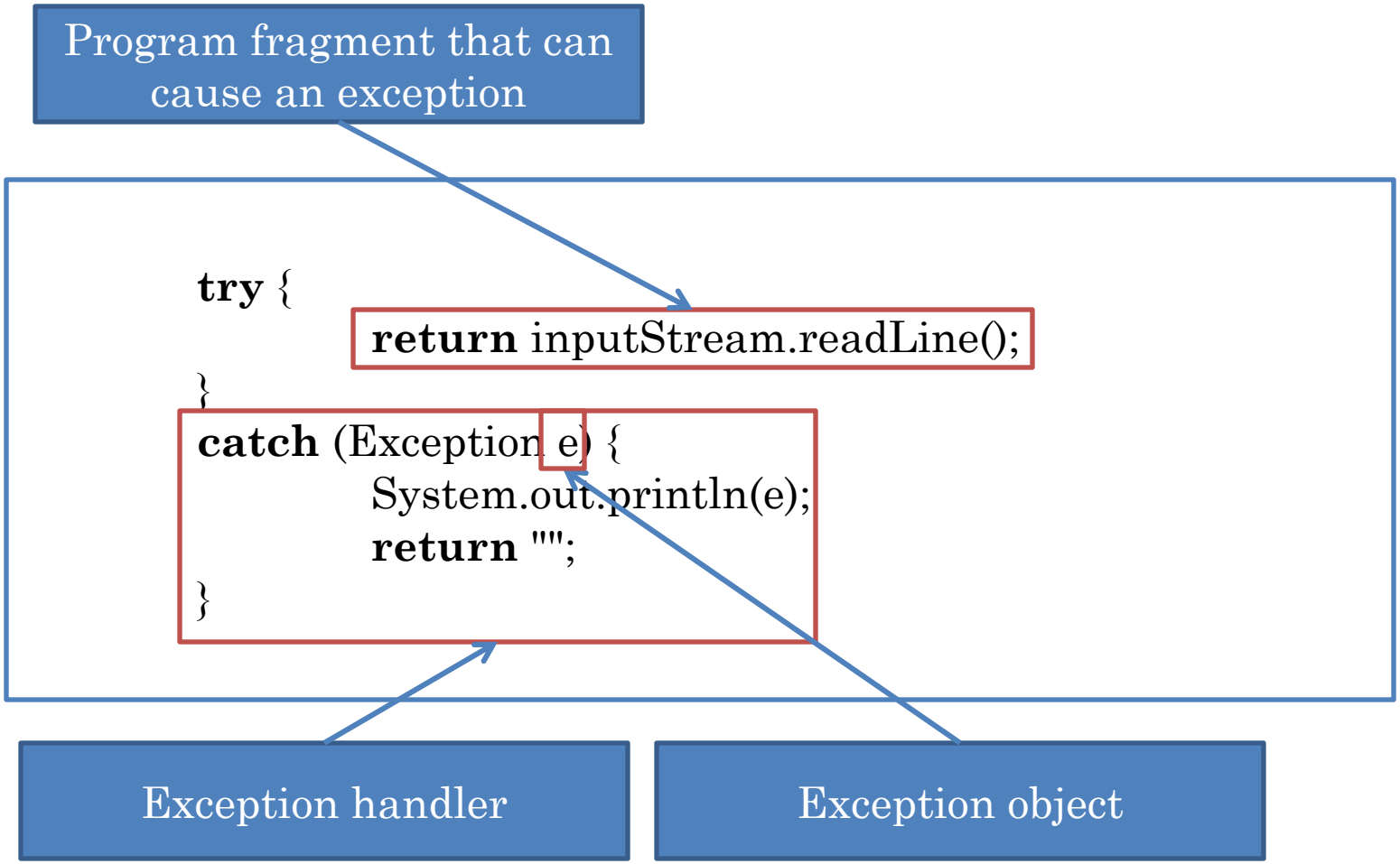
While Loops

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product = product* nextNum;
        nextNum = Console.readInt();
}
System.out.print (product);
```

Input Class

```
public class Console {
   static BufferedReader inputStream =
      new BufferedReader(new InputStreamReader(System.in));
   public static int readInt() {
            try {
                     return Integer.parseInt(inputStream.readLine());
            } catch (Exception e) {
                     System.out.println(e);
                     return 0;
            }
   }
   public static String readString() {
            try {
                     return inputStream.readLine();
            } catch (Exception e) {
                     System.out.println(e);
                     return "";
            }
   }
  ... //other methods
}
```

Exceptions and Try-Catch Block



For Loops, Arrays and Comments

```
System.out.println("Number of Strings:");
int numElements = Console.readInt(); // reads the next line as integer
System.out.println("Please enter " + numElements + " strings");
String[] strings = new String[numElements]; // dynamic array
for (int elementNum = 0;  elementNum < numElements; elementNum++)
        strings[elementNum] = Console.readString();
```

```
/* This loop uses the array input
** in the previous loop*/
for ( int elementNum = 0; elementNum < strings.length; elementNum++)
        System.out.println(strings[elementNum]);
```

```
String s =  strings[0]; // unsafe
for (int i=0; i<s.length(); i++)
        System.out.println(s.charAt(i));
```

Difference in syntax: arrays built into language, strings are library

Accessing Substrings

s.substring(beginIndex, endIndex)

"hello world".substring(4,7) → "o w"

"hello world".substring(4,4) → ""

"hello world".substring(7,4)

StringIndexBounds exception

METHODS/PROCEDURES/FUNCTIONS

```
<terminated> Factorial [Ja
3
factorial = 6
2
factorial = 2
-1
```

1*2*3*…*n

```
static int f (int n) {
        int product = 1;
        while (n > 0) {
                product *= n;
                n -= 1;
        }
        return product;
}
```

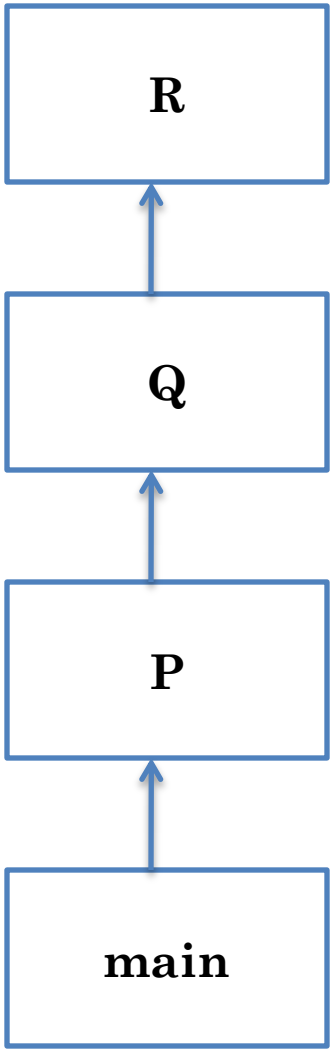
- Called method
- Takes int argument, n,
- Returns int

```
public static void main (String[] args) {
        while (true) {  // loop condition never false
                int n = Console.readInt();
                if (n < 0)
                        break;
                System.out.println("factorial = " + f(n));
        }
}
```

- Calling method.
- Takes String array argument
- Returns nothing – void

Static implies non-object oriented programming.

Call Chains



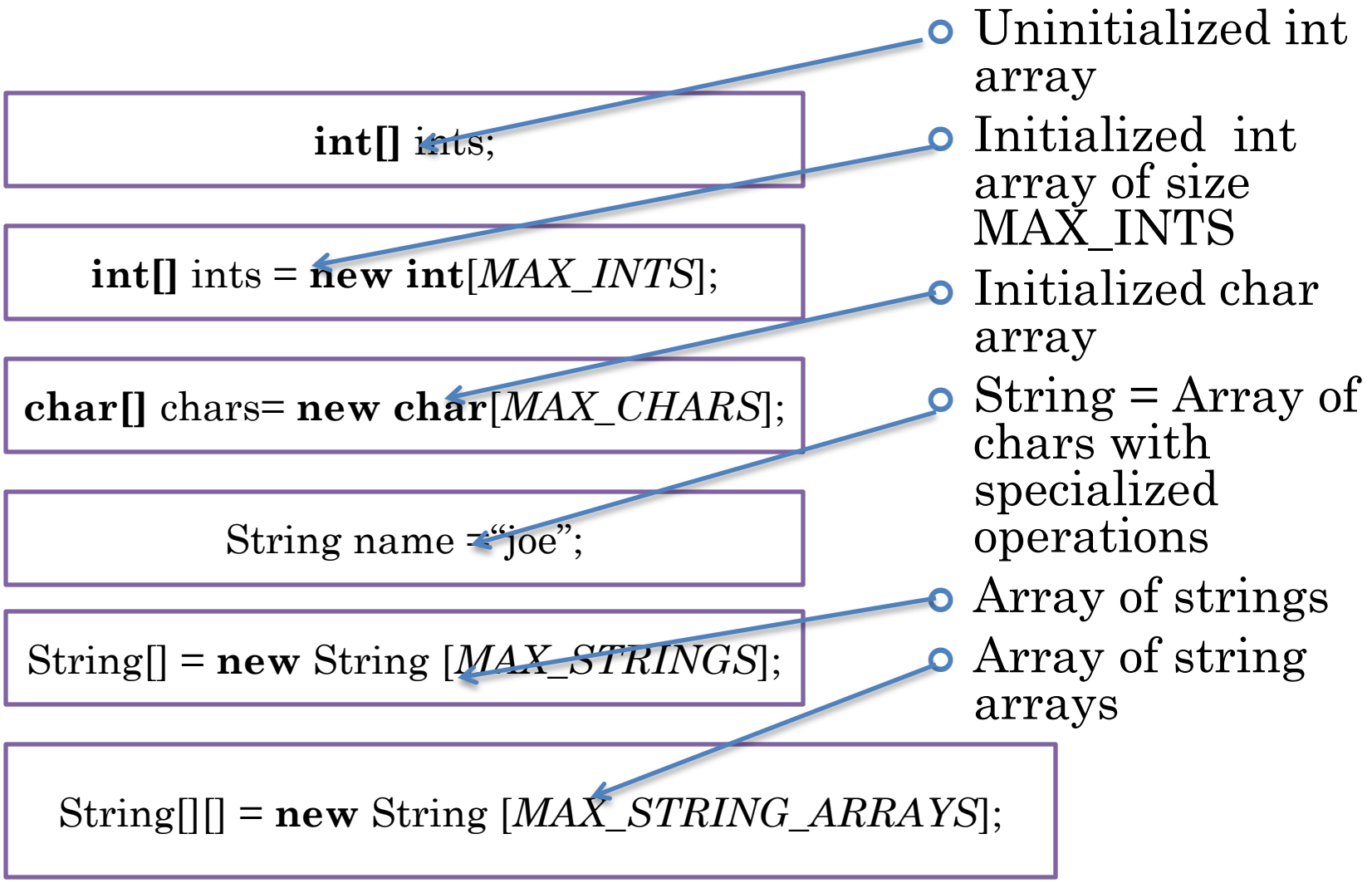
Main method starts the computation, and can call other methods.

Can put complete program in main method

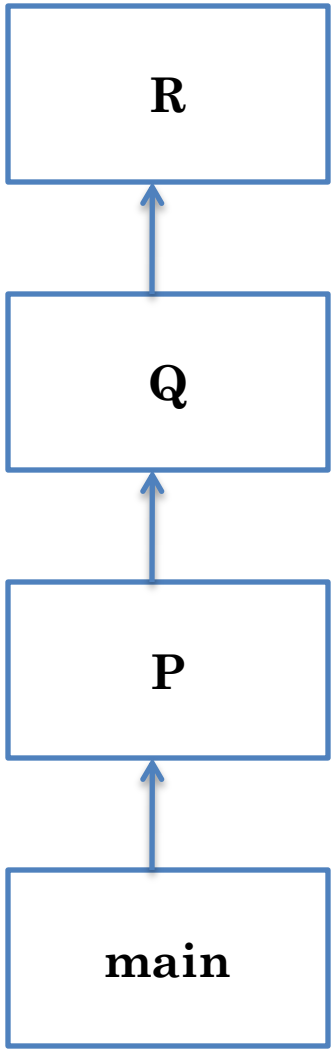
Like having one big paragraph in an essay

Method decomposition important modularization technique even in conventional programming

Arrays


int[] ints;
int[] ints = new int[MAX_INTS];
char[] chars= new char[MAX_CHARS];
String name =“joe”;
String[] = new String [MAX_STRINGS];
String[][] = new String [MAX_STRING_ARRAYS];
Uninitialized int array
Initialized int array of size MAX_INTS
Initialized char array
String = Array of chars with specialized operations
Array of strings
Array of string arrays

Main Method Details



Main method has predefined header.

```
public static void main (String[] args) {
        ….
}
```

All methods must be in some "class" (file, which can be in a "package" (directory)

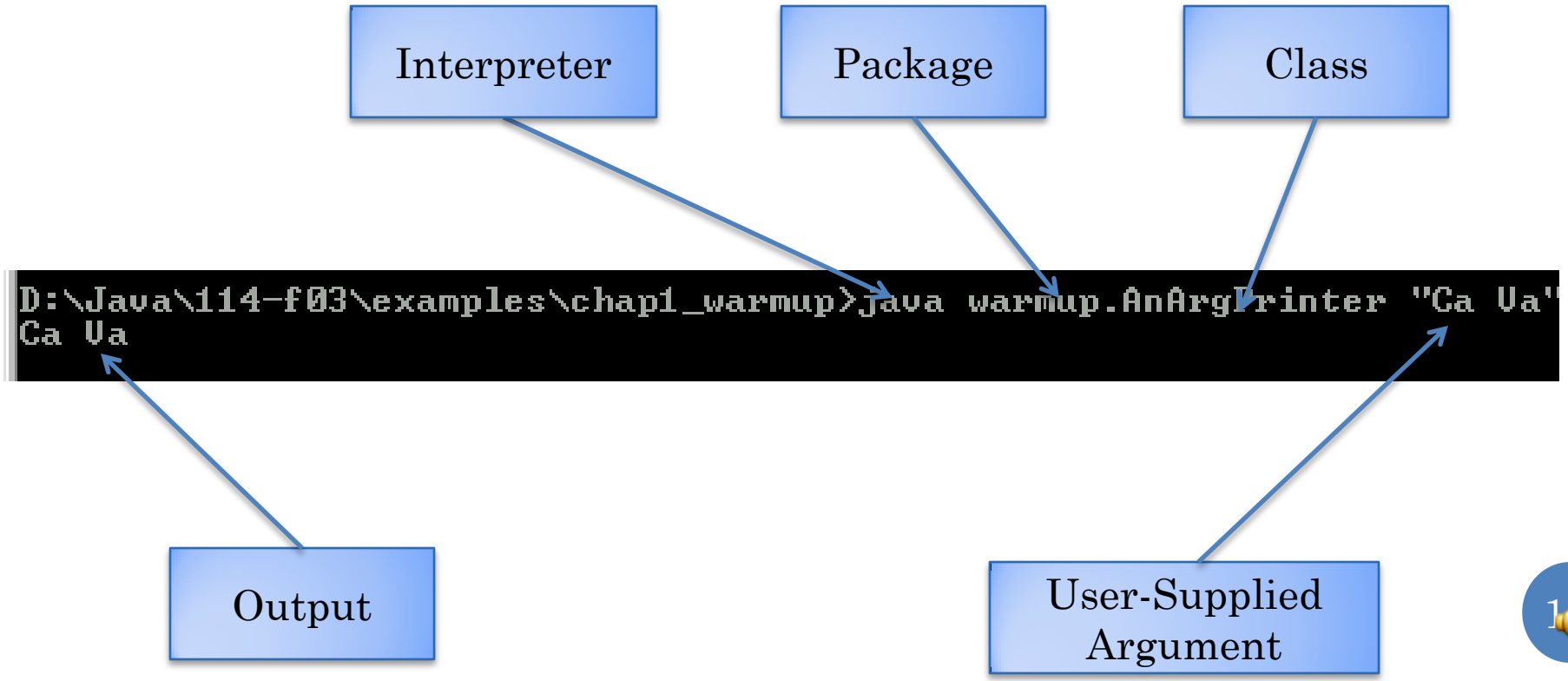
```
package warmup;
public class AnArgPrinter {
    public static void main (String[] args) {
        System.out.println (args[0]);
    }
}
```

The Java interpreter calls main and provides its user-specified argument. Public means interpreter can access main.

Running Main Class

```
package warmup;
public class AnArgPrinter {
    public static void main (String[] args) {
        System.out.println (args[0]);
    }
}
```

Array of user-supplied strings



Array Subscript Error

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

```
D:\Java\114-f03\examples\chap1_warmup>java warmup.AnArgPrinter
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at warmup.AnArgPrinter.main(AnArgPrinter.java:7)
```

Subscript Error

User-Supplies No Argument

Safe Arg Printer

```
D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter "Ca Va"
Ca Va

D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter
Illegal no of arguments:0. Terminating program

D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter "Ca Va" "Namas
te"
Illegal no of arguments:2. Terminating program
```

Safe Arg Printer (edit in class)

```
package warmup;
public class AnArgPrinter {
    public static void main (String[] args) {
        System.out.println (args[0]);
    }
}
```

```
D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter "Ca Va"
Ca Va

D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter
Illegal no of arguments:0. Terminating program

D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter "Ca Va" "Namas
te"
Illegal no of arguments:2. Terminating program
```

Safe Arg Printer

```
package warmup;
public class AnArgPrinter {
   public static void main (String[] args) {
     if (args.length == 1)
          System.out.println (args[0]);
      else
          System.out.println("Illegal no of arguments:" + args.length + ".
Terminating program");
     }
}
```

String concatenation

```
D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter "Ca Va"
Ca Va

D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter
Illegal no of arguments:0. Terminating program

D:\Java\114-f03\examples\chap1_warmup>java warmup.ASafeArgPrinter "Ca Va" "Namas
te"
Illegal no of arguments:2. Terminating program
```

Class with Multiple Methods

```
public class FactorialPrinter {
      static int f (int n) {
            int product = 1;
            while (n > 0) {
                  product *= n;
                  n -= 1;
            }
            return product;
       }
      public static void main (String[] args) {
            while (true) {   // loop condition never false
                  int n = Console.readInt();
                  if (n < 0)
                        break;
                  System.out.println("factorial = " + f(n));
            }
}
```

Monolithic Method

```
System.out.println("Number of Strings:");
int numElements = Console.readInt(); // reads the next line as integer
System.out.println("Please enter " + numElements + " strings");
String[] strings = new String[numElements]; // dynamic array
for (int elementNum = 0;  elementNum < numElements; elementNum++)
        strings[elementNum] = Console.readString();
```

```
/* This loop uses the array input
** in the previous loop*/
for ( int elementNum = 0; elementNum < strings.length; elementNum++)
        System.out.println(strings[elementNum]);
```

```
String s =  strings[0]; // unsafe
for (int i=0; i<s.length(); i++)
        System.out.println(s.charAt(i));
```

Boxes and comments show that there are multiple logically separate steps

How can we break this code into different methods

Method Decomposition

```
public static void forArrayCommentsWithMethodDecomposition() {
  String[] strings = readStrings(readNumStrings());
  printStrings(strings);
  printString(strings[0]);// unsafe
}
```

```
public static int readNumStrings() {
  System.out.println("Number of Strings:");
  return Console.readInt(); // reads the next line as integer
}
```

```
public static String[] readStrings(int numElements) {
  System.out.println("Please enter " + numElements + " strings");
  String[] strings = new String[numElements]; // dynamic array
  for (int elementNum = 0; elementNum < numElements; elementNum++)
    strings[elementNum] = Console.readString();
  return strings;
}
```

Method Decomposition (Contd.)

```
public static void printStrings(String[] strings) {
  for (int elementNum = 0; elementNum < strings.length;
           elementNum++)
System.out.println(strings[elementNum]);
}
```

```
public static void printString(String s) {
  for (int i = 0; i < s.length(); i++)
    System.out.println(s.charAt(i));
}
```