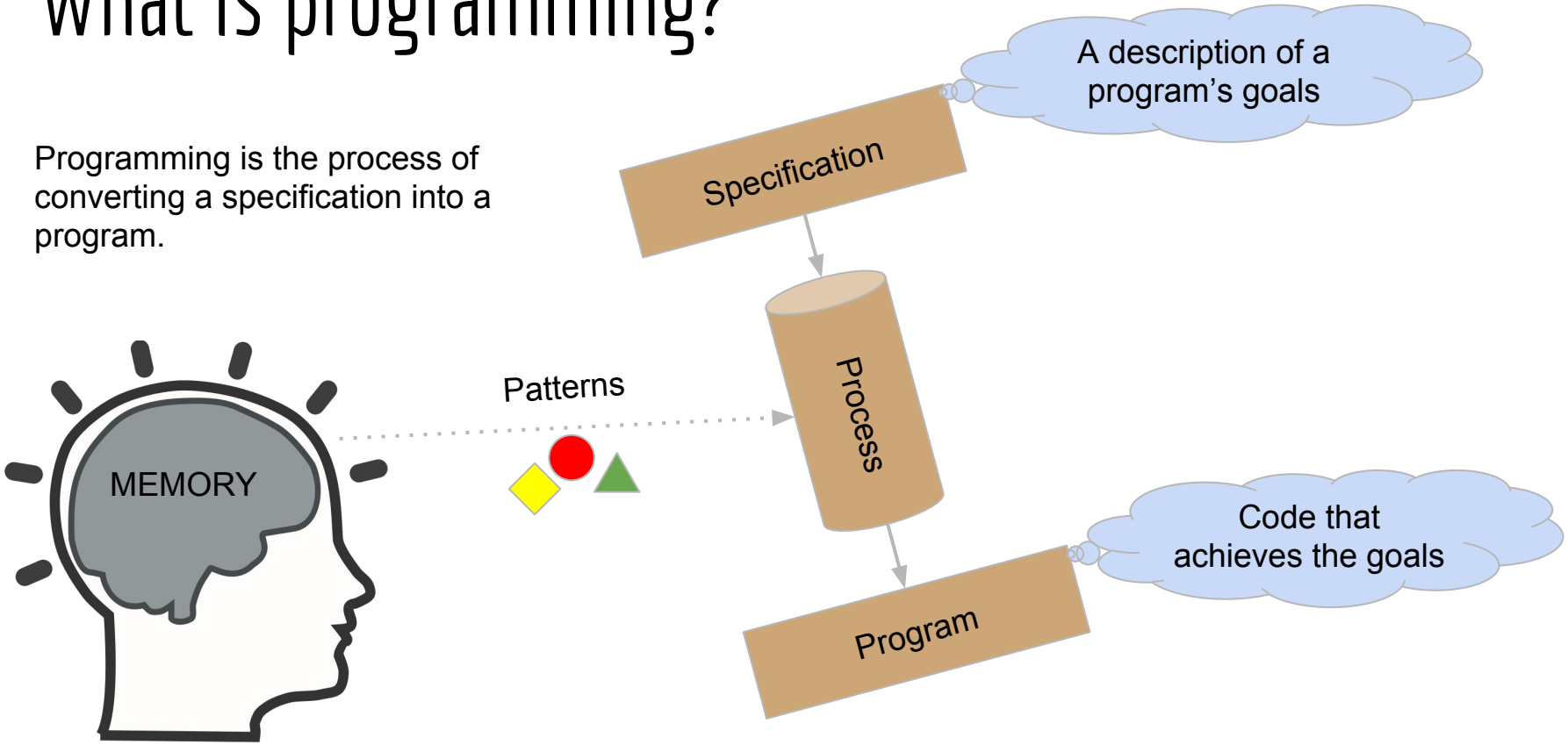
Week 1

Basic Patterns

What is programming?

Programming is the process of converting a specification into a program.



What is the process?

Not a single process!

- Words to code table
- Key/framework approach
- Debugging/Testing
- Break it down, build it up
- Class diagrams
- Location tables
- + a bit of creativity and intuition[1]

[1] A classic text on creativity: “Lateral Thinking” by Edward De Bono (1970)

What is a pattern?

- A common solution to a common problem.

Example program:

```
int totalRainfall = 0;
for (int i = 0; i < rainfall.length; i++)
{
    totalRainfall += rainfall[i];
}
```

The pattern: “sum”

```
sum = 0;
for each item
{
    sum += item;
}
```

The “sum” pattern

Goal: Find the sum of a collection of items.

```
<type> sum = 0;
<for each item>
{
    sum += <item>;
}
```

- Words enclosed between `<angled brackets>` are holes that need to be filled.
- `<type>` is typically double or int
- `<for each item>` is any loop over a collection of items
- `<item>` refers to the next item in the loop.
- `x += y` adds the amount y onto x.

The “output” pattern

Goal: Show a value to the user.

- **Pattern:**

```
System.out.println(“<label>” + <value>);
```

- **e.g. show an age:**

```
System.out.println(“age is “ + age);
```

- **e.g. show a name:**

```
System.out.println(“name is “ + name);
```

The “read” pattern

Goal: Read a value from the user.

- **Pattern:**

```
System.out.print(“<prompt>”);
<type> <variable> = <read operation>;
```

Or,

```
System.out.print(“<prompt>”);
<variable> = <read operation>;
```

(if <variable> has already been declared)

- **e.g. read an age:**

```
System.out.print(“Age: “);
int age = In.nextInt();
```

Read operations

- We provide you with class `In` which has 4 static methods:

public static int nextInt()	Reads an integer
public static double nextDouble()	Reads a double
public static char nextChar()	Reads a single character
public static String nextLine()	Reads a String

- Copy this class into your project.
  Then simply call the method on the class: `In.nextInt()`

The “read loop” pattern

Goal: Read values until the user enters an “end of input” value.

```
<read pattern>
while (<value> != <end value>)
{
    <use the value>
    <read pattern>
}
```

Observations:

- <read pattern> appears twice
- always test for the “end of input” value immediately after a <read pattern>.

The “array loop” pattern

Goal: Loop over items in an array.

```
for (int i = 0; i < <array>.length; i++)
{
    <use the item array[i]>
}
```

Example 1: Calculate total rainfall

```
double[] rainfall = { 102.5, 117.0, 129.2, 128.6, 119.9,
                      131.9, 97.0, 80.6, 68.4, 76.7, 84.1, 77.7 };
double sum = 0.0;
for (int i = 0; i < rainfall.length; i++)
{
    sum += rainfall[i];
}
System.out.println("Total rainfall = " + sum);
```

Patterns used:

- sum
- array loop
- output

Example 2: Calculate total value of cards

```
int sum = 0;
System.out.print("Enter card number: ");
int card = In.nextInt();
while (card != -1) {
    sum += card;
    System.out.print("Enter card number: ");
    card = In.nextInt();
}
System.out.println("Total card value = " + sum);
```

Patterns used:

- sum
- read
- read loop
- output

The “count” pattern

Goal (without guard): Count the number of items in a collection.
Goal (with guard): Count the number of items that satisfy a condition.

Without guard:

```
int count = 0;
<for each item>
    count++;
```

With guard:

```
int count = 0;
<for each item>
    if (<guard>)
        count++;
```

X++ VS ++X

- Both x++ and ++x add 1 to x.
- x++ returns the value of x and THEN adds 1 to x.
- ++x adds 1 to x and THEN returns the value of x.

e.g.

```
int x = 7;
System.out.println(x++); // shows
7
                    x = 8
```

```
int x = 7;
System.out.println(++x); // shows
8
                    x = 8
```

Example 3: Count royalty cards

```
int count = 0;
System.out.print("Enter card number: ");
int card = In.nextInt();
while (card != -1) {
    if (card > 10)
        count++;
    System.out.print("Enter card number: ");
    card = In.nextInt();
}
System.out.println("Number of royalty cards = " + count);
```

Patterns used:

- count
- read
- read loop
- output

The “max” pattern

Goal: Find the maximum value in a collection of items

```
<type> max = <smallest number>;
<for each item>
{
    if (<item> > max)
        max = <item>;
}
```

Key idea:

If this item is bigger than the max so far, then make it the new max.

Process

Process: Words-to-code table

Specification: Read in card numbers until the user enters -1. Show the highest card.

```
Enter card number: 8
Enter card number: 3
Enter card number: 12
Enter card number: 7
Enter card number: -1
The highest card is 12
```

Process: Words-to-code table

Specification: Read in card numbers until the user enters -1. Show the highest card.

Break the specification down into pieces:

Read in the card numbers /

until the user enters -1 /

Show /

the highest card

Process: Words-to-code table

Arrange the words into a table:

Words	**Code/pattern**
Read in card numbers	
until the user enters -1	
Show	
the highest card	

Process: Words-to-code table

Translate the words to code:

Words	**Code/pattern**
Read in card numbers	read
until the user enters -1	read loop
Show	output
the highest card	max

Solution

```
int max = Integer.MIN_VALUE;
System.out.print("Enter card number: ");
int card = In.nextInt();
while (card != -1) {
    if (card > max)
        max = card;
    System.out.print("Enter card number: ");
    card = In.nextInt();
}
System.out.println("Highest card = " + max);
```

Patterns used:

- max
- read
- read loop
- output