Programming Using Java

Session 2: Building Algorithms

Problem Solving using Computers

- computers execute programs to provide solutions to problems
- programs are implementations of algorithms
- algorithms are composed of….
  - a sequence of actions
  - an order of execution for those actions
- execution involves accepting input…and generating output

Building Algorithms

- use prose for documenting
- actions usually include input, output or computation operations
- control structures determine order of execution
- actions cannot be ambiguous

Example: Mobile Phone

Posting a Tweet From a Mobile Phone

1. check to see if phone is locked
2. if phone locked, enter security code, otherwise go to step 6
3. increment record number of attempts
4. check to see if phone is locked
5. if phone is locked
    - if maximum number of attempts is exceeded, STOP
    - otherwise go to step 1
6. take picture
7. login to Twitter
8. post tweet
9. STOP

Posting a Tweet From a Mobile Phone

1. check to see if phone is locked
2. **if** phone locked, enter security code, **otherwise go to** step 6
3. increment record number of attempts
4. check to see if phone is locked
5. **if** phone is locked
    - **if** maximum number of attempts is exceeded, STOP
    - **otherwise go to** step 1
6. take picture
7. login to Twitter
8. post tweet
9. STOP

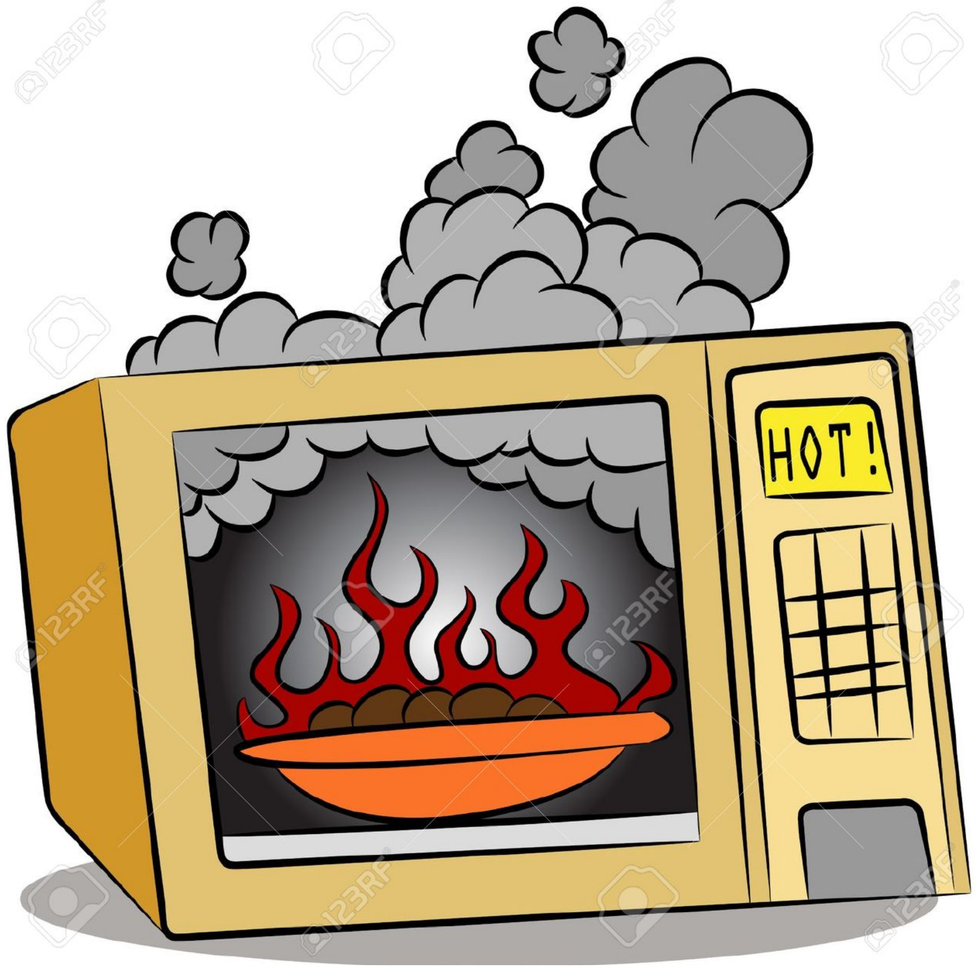
Control Structures

- all algorithms can be expressed using the following control structures

  1. sequence
  2. selection
  3. iteration

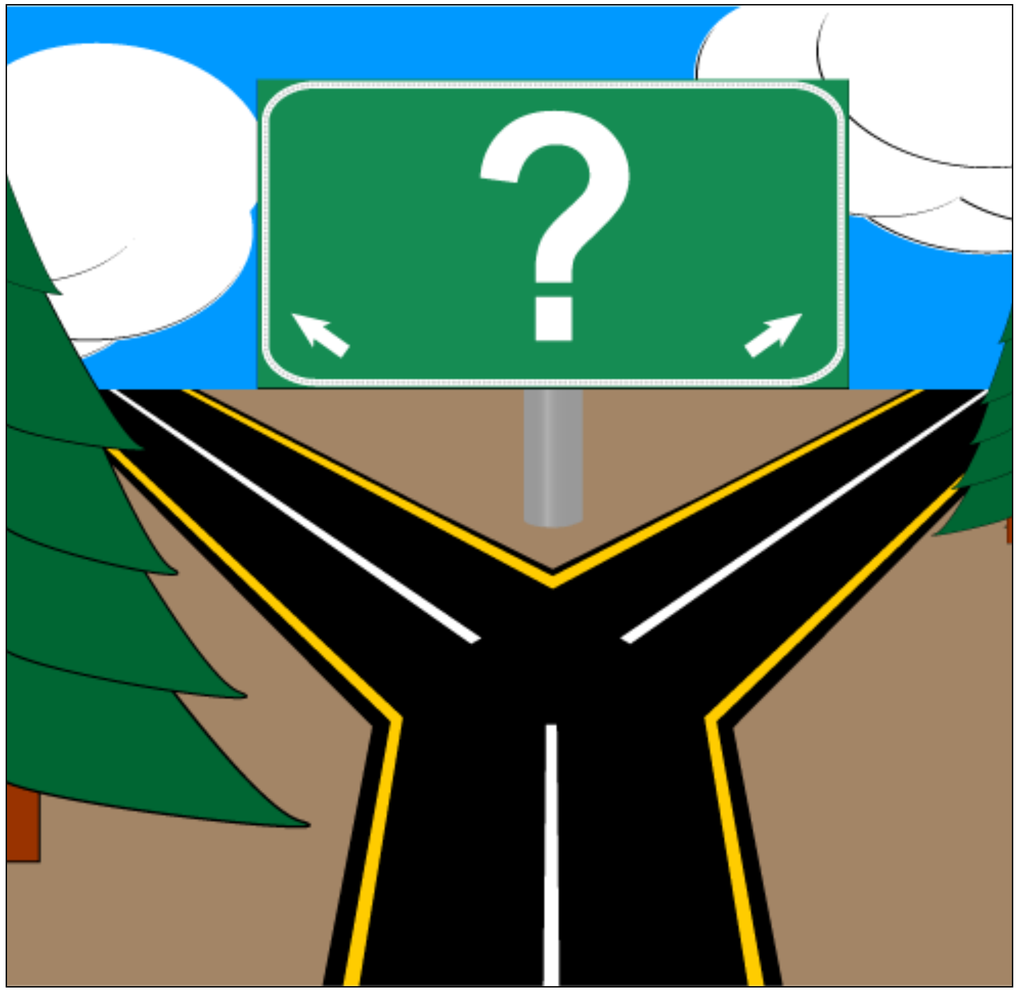
Sequence Structure

- multiple commands
- all actions executed in order presented
- actions are executed exactly once
- “do action A, then action B…”



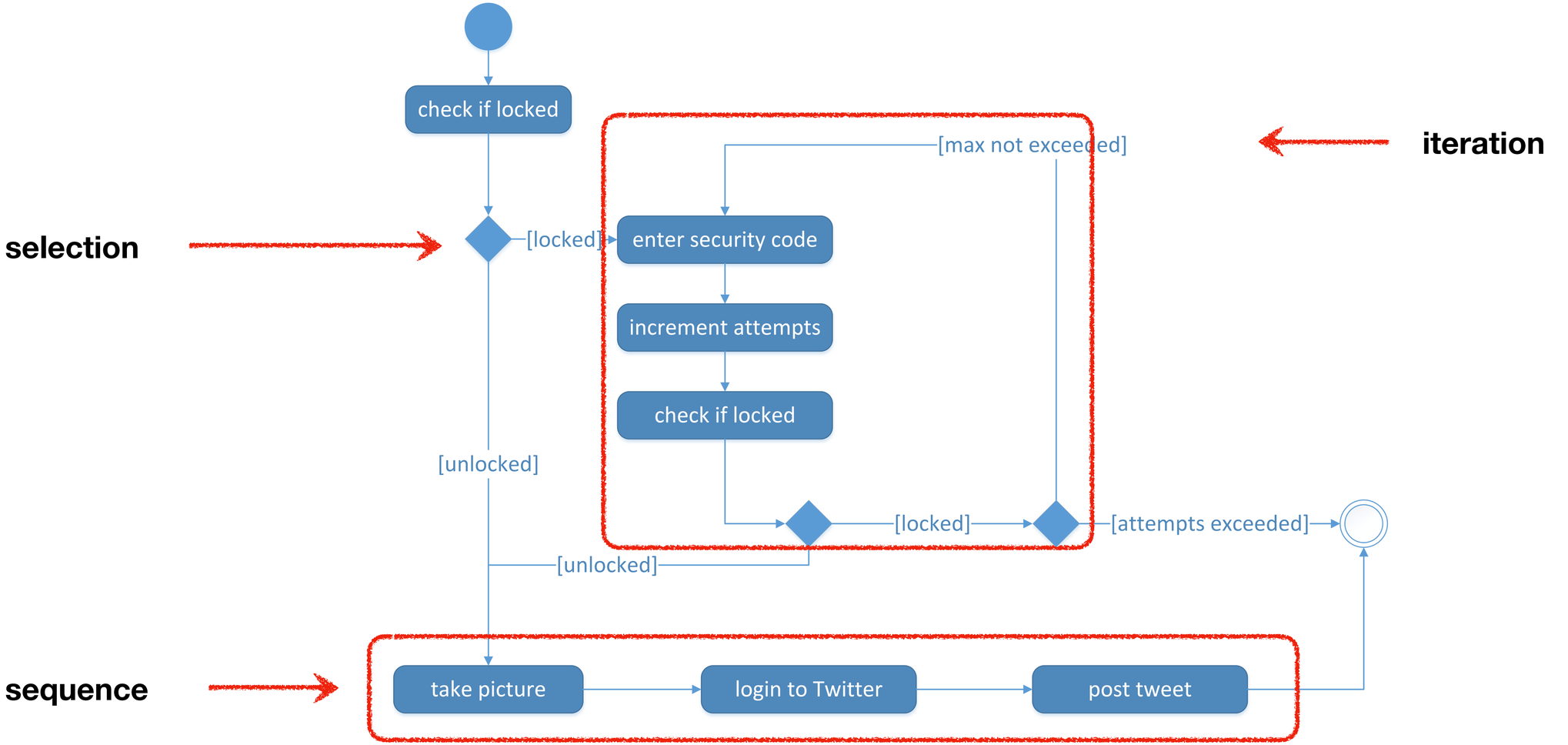
Selection Structure

- choice of action taken depends on a condition
- an action may or may not be executed
- “**if condition** holds **then** do action A **otherwise** do action B…”
- the otherwise clause can have no action



Iteration Structure

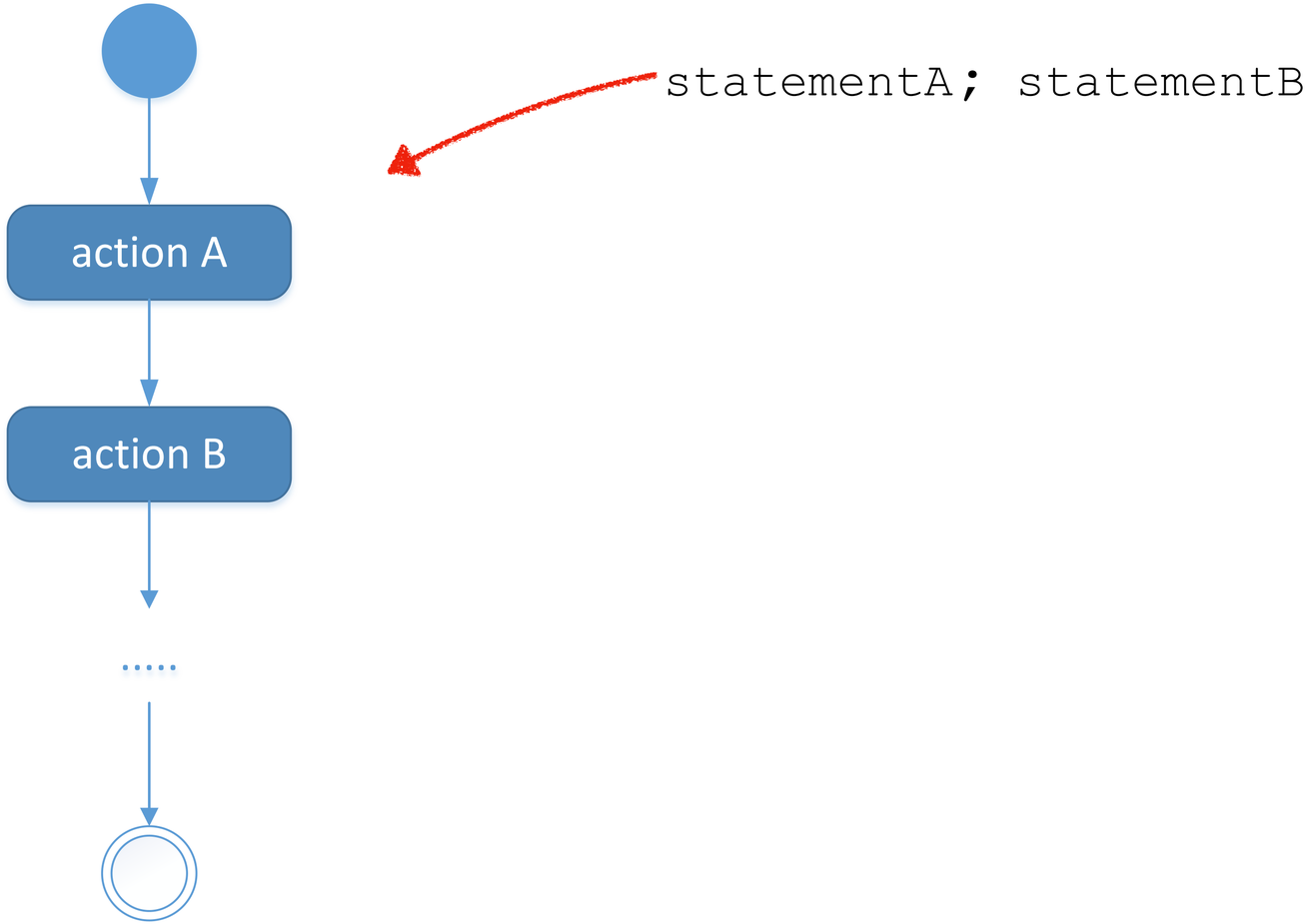
- allows actions to be executed more than once
- can repeat actions
  - a specific number of times
  - until a condition holds
- “while condition holds do action A…”



Structure Hierarchy

"…control structures are combined to build complex algorithms…"

Control Structures in Java

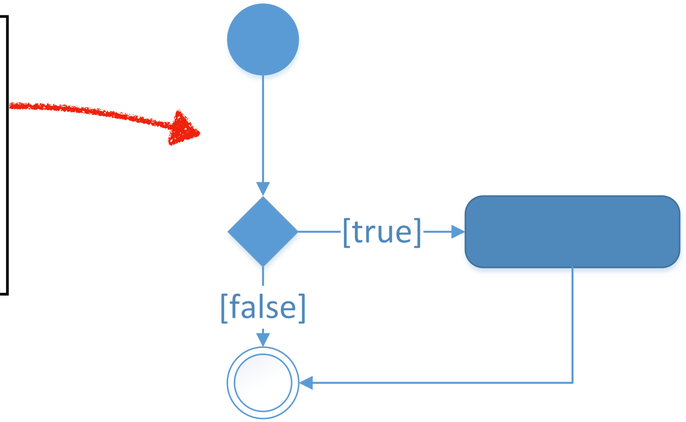


Sequence Structure

"…statements executed in order of appearance…"

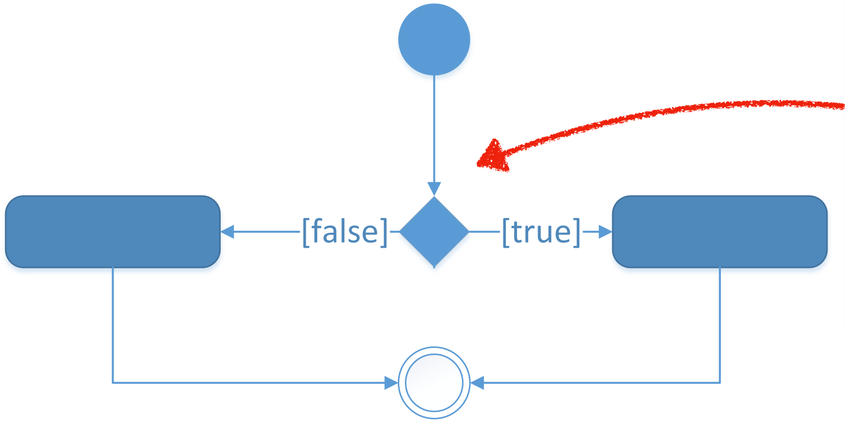
if statement

```
if (cond)
  statement;
```



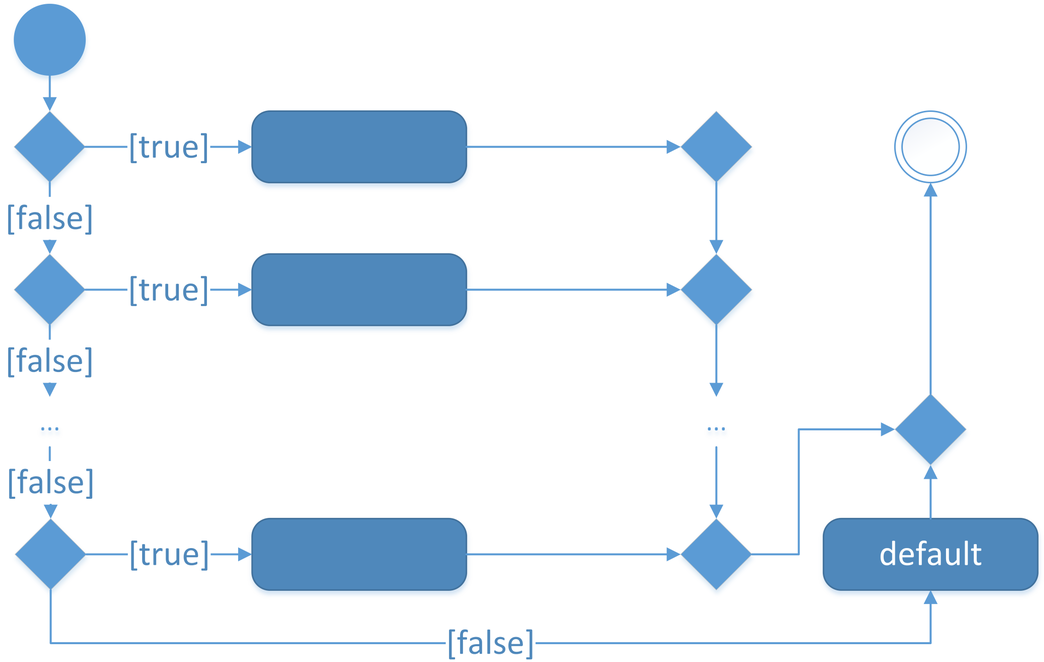
if - else statement

```
if (cond)
  statementA
else
  statementB
```



switch statement

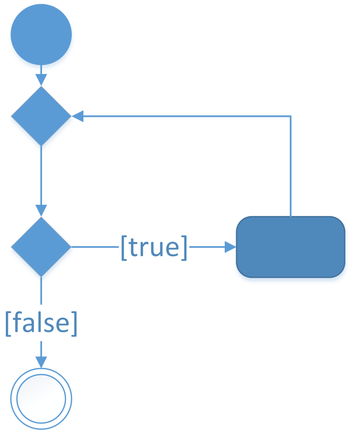
```
switch (getState()) {
 case state1:
  statementA; break;
 case state2:
  statementB; break;
 default;
  statementC; break;
}
```

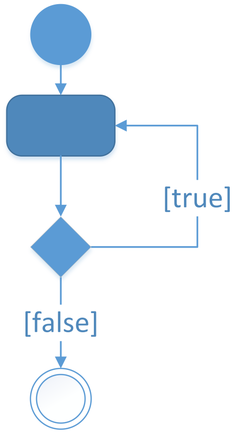


Conditional Structures

while statement

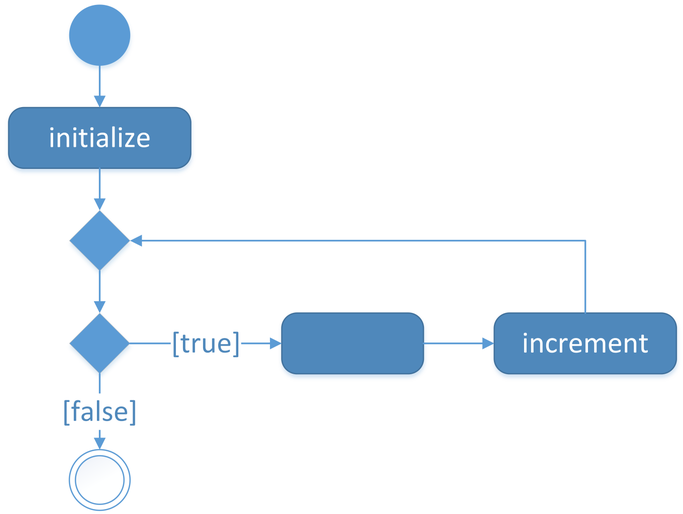
```
while (cond)
  statement;
```





do-while statement

```
do {
 statement;
} while(cond);
```



for statement

```
for (initialization,
     continueCond,
     increment)
 statement;
```

Iteration Structures

Symbol	Operator	
&&	conditional AND	binary
||	conditional OR	binary
!	logical negation	unary
==	equality	binary
!=	inequality	binary
<	less than	binary
>	greater than	binary

Logical Operators

"…take operands and return a boolean result…"

Symbol	Operator	
*	multiplication	binary
/	division	binary
+	addition	binary
-	subtraction	binary

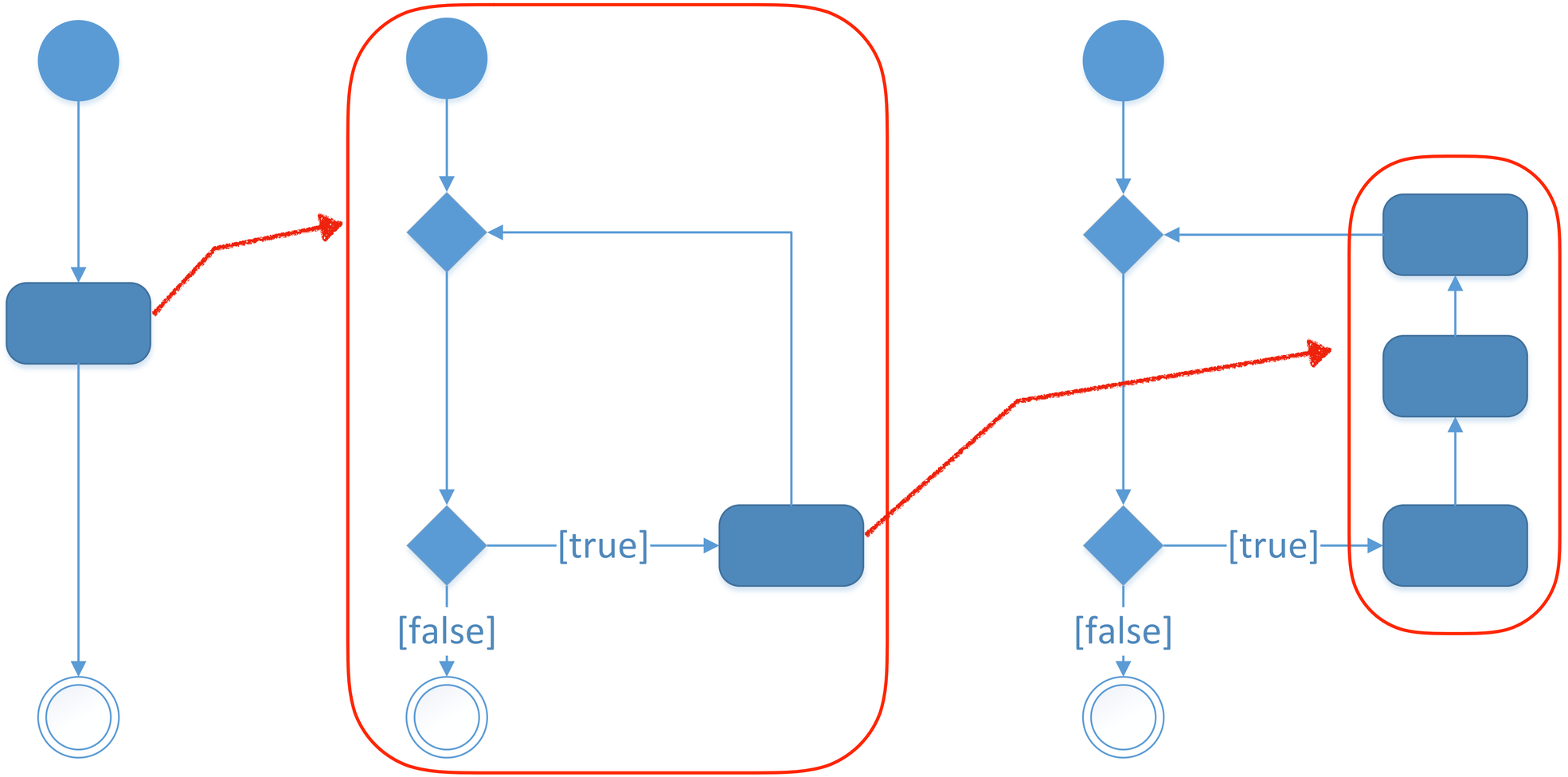
Mathematical Operators

"…take operands and return a numeric result…"

Rules for Building Programs	
1	Begin with a simple diagram
2	Replace an action with 2 actions in sequence
3	Replace an action with a control structure
4	Apply rules 2 and 3 until done

Structured Program Generation

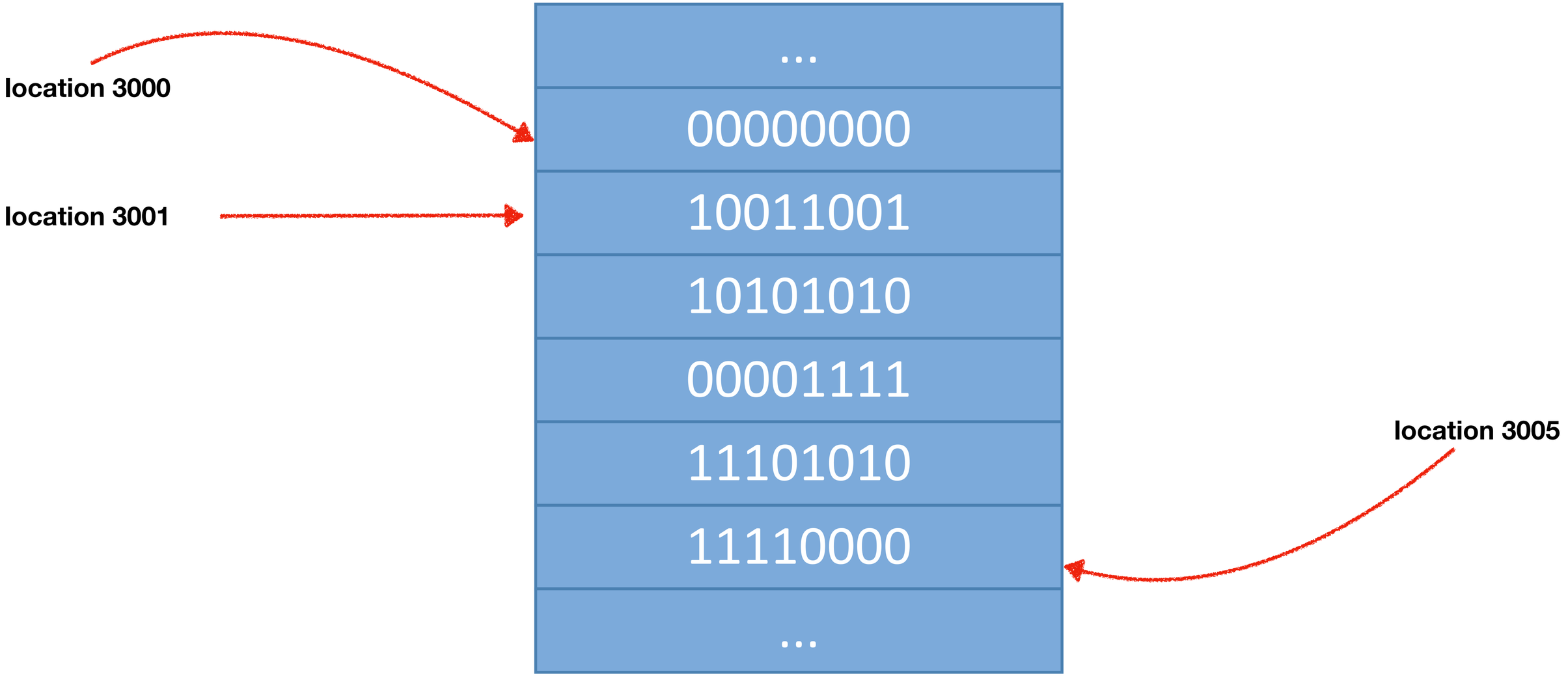
"…refine your algorithm until it can be represented as a program…"



Stages of Program Refinement

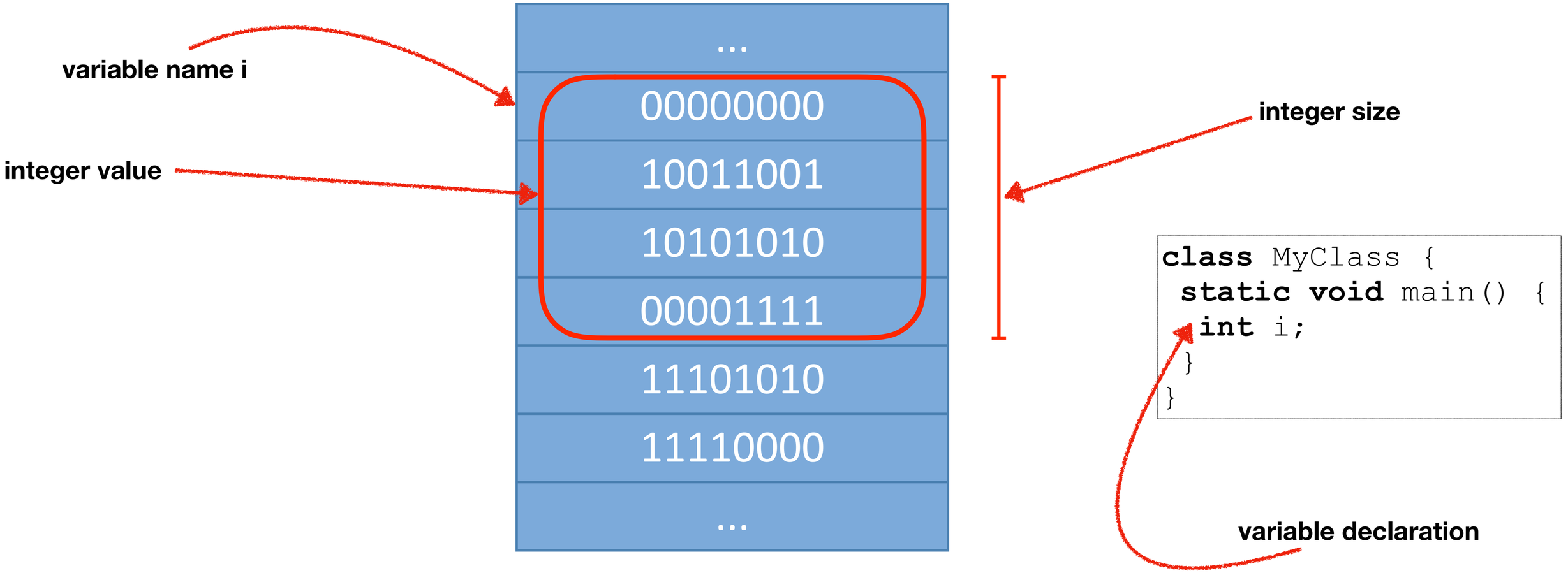
"…add more detail to algorithm description…"

Using Memory for Java Programs



Memory Layout

"…a sequence of bits partitioned into blocks…"

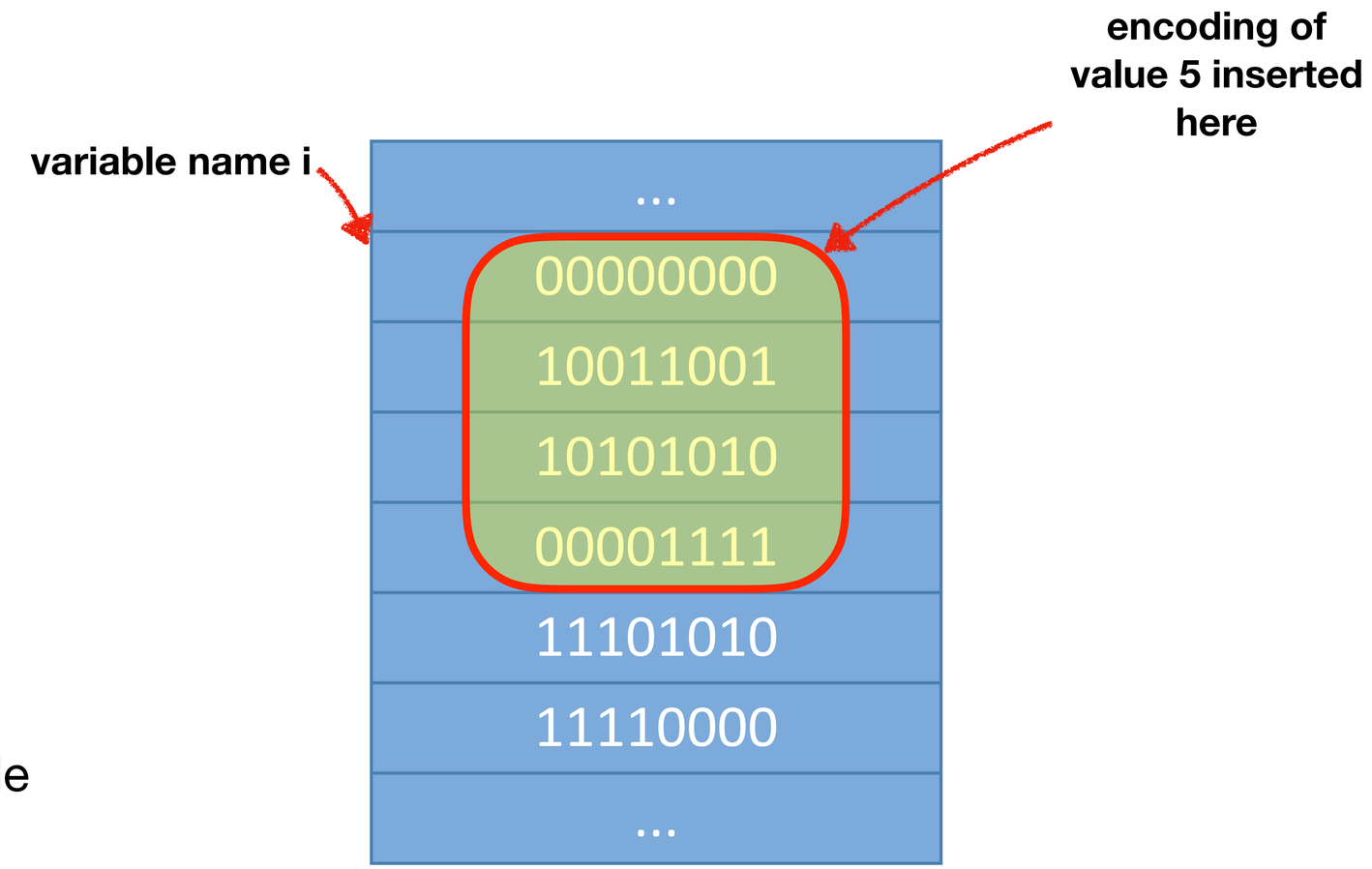


Variables

"…point to a location in memory…"

Assignment Statement

```
class MyClass {
 static void main() {
  int i;
  i = 5;
  i = Integer.parseInt("5");
  i = 5 * 1;
 }
}
```

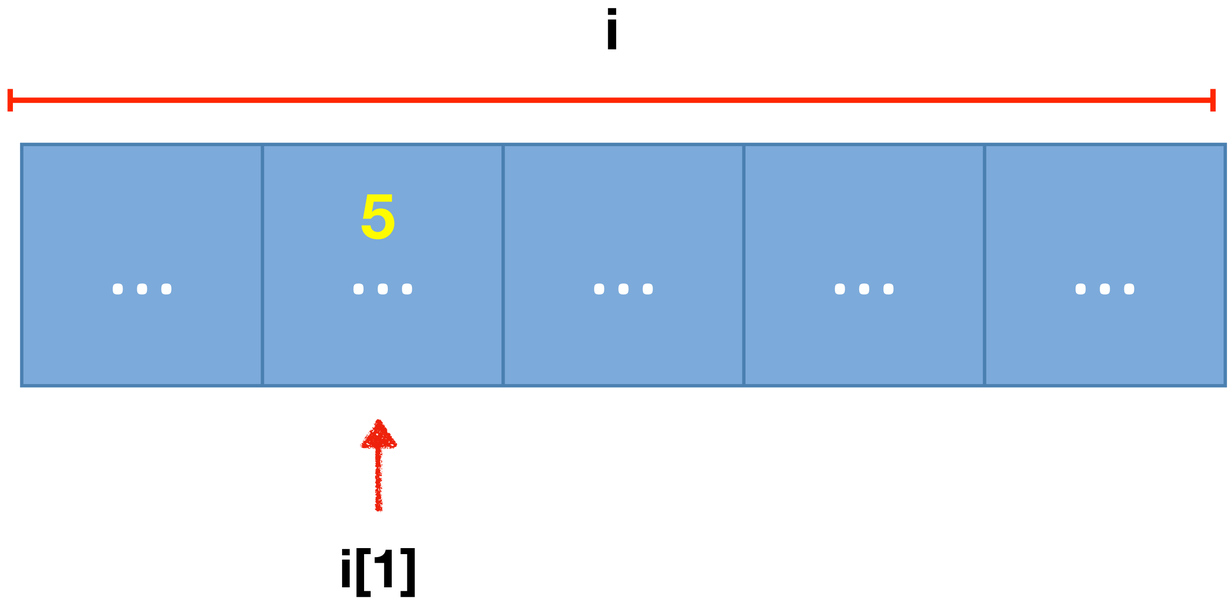


- a value is placed into memory location pointed to by variable name
- value must be of the same type as variable
- conversion functions are available

Arrays

```
class MyClass {
 static void main() {
  int i[] = new int[5];
  i[1] = 5;
 }
}
```

- ordered group of elements of the same type
- fixed number of elements
- elements can be accessed by index (zero-based)



Arrays

```
public class ArraySample {

  public static void main(String[] args) {

    // initializing with static values
    double arr[] = {1,2,3};
    double total = 0;

    // iterating through all elements
    for (int i=0; i < arr.length; i++)
      total = total + arr[i];

    // a more compact way of traversal
    for (double c : arr)
      total = total + c;
   }
}
```