Java Basics – Part 2

Basic Java

- Array
- Loops and control

Basic Java - Array

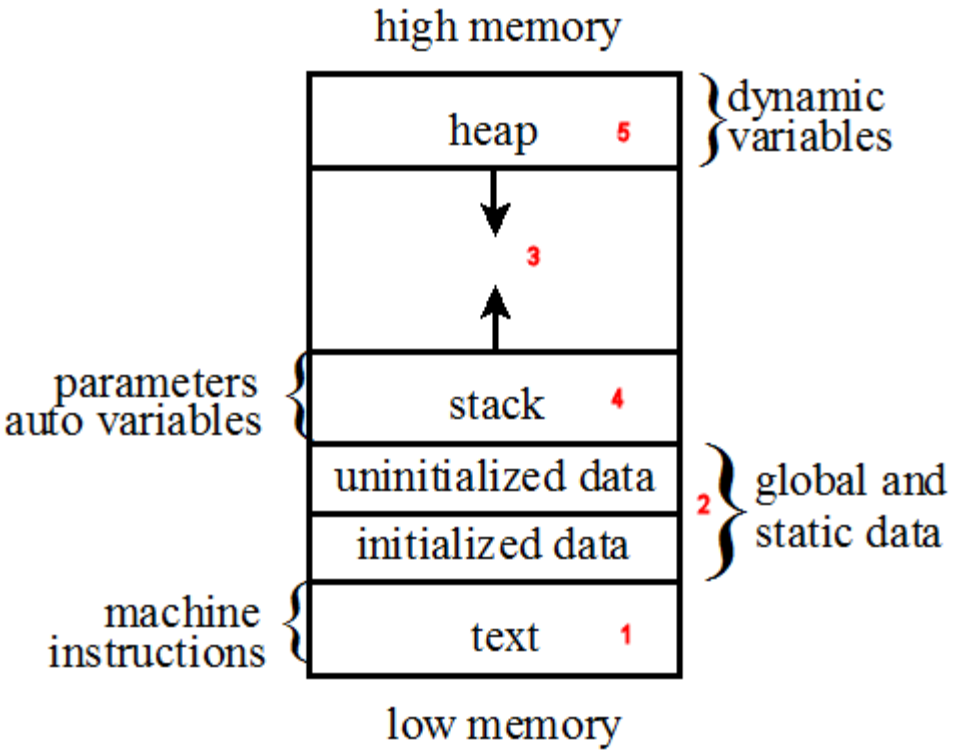
- Java has built in arrays, that hold elements of the same type - primitive data types or classes

- space for array must be dynamically allocated with *new* operator.

```
// declare an array without size given
double[] myDoubleArray;

// define an array with fixed size
int[] myIntArray = new int[10];
float myFloatArray[] = new float[20];
```

Basic Java - Array

- What does the 'new' exactly mean?
  - New object
  - New memory space allocation




```
// declare an array without size given
double[] myDoubleArray;

// define an array with fixed size
int[] myIntArray = new int[10];
float myFloatArray[] = new float[20];
```

http://icarus.cs.weber.edu/~dab/cs1410/textbook/4.Pointers/memory.html

Basic Java - Array

- arrays have a public, final field called *length*
- elements start with an index of zero, last index is length - 1

```
float myFloatArray[] = new float[20];

// print the length
System.out.prinln(myFloatArray.length);

// print the first and last element
System.out.prinln(myFloatArray[0]);

System.out.prinln(myFloatArray[19]);
```

Basic Java - Array

- What happens if you do below?

```
float myFloatArray[] = new float[20];

System.out.prinln(myFloatArray[20]);
```

Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException

Basic Java - Array

- Initialization
  - by default, all values in the array are initialized (0, 0.0, char 0, false, or null)

```
// option one
int myIntArr[] = new float[3];
System.out.println(myIntArr[1]);

// modify the vaue
myIntArr[0] = 1; myIntArr[1] = 2;
myIntArr[2] = 3;


// option two
int myIntArr[] = {1, 2, 3};
boolean myBooArr[] = {true, false, false};
```

Basic Java - Array

- What happens if you do below?

```
float myFloatArray[];

myFloatArray[0] = 5.5;
```

Error message:

variable myFloatArray might not have been initialized

- But you can do the following without problem … why?

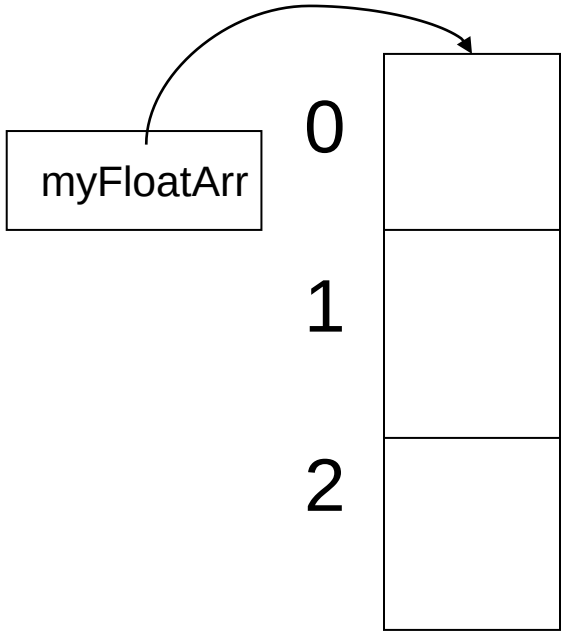
```
float myFloat;

myFloat = 5.5;
```

Basic Java - Array

- The reason is that array variables are object variables, and they hold the memory address of an array object
  - Like a pointer

```
// pointing to null
float myFloatArr[];
myFloatArray[0] = 5.5;
```

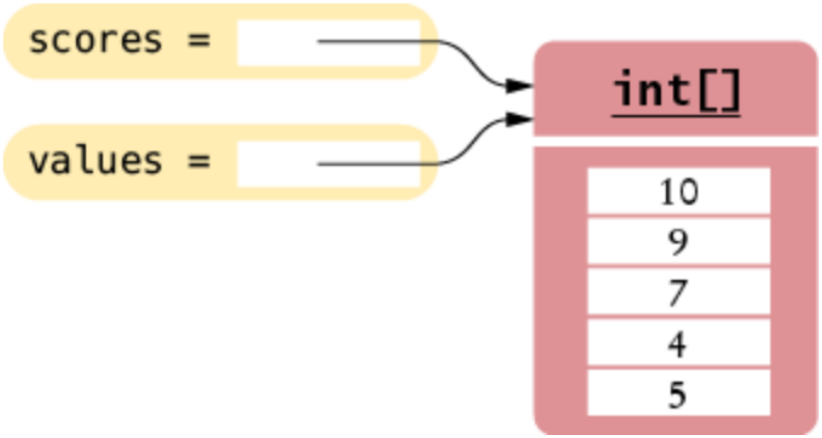
```
// pointing to 1st element
float myFloatArr = new float[3];
myFloatArr[0] = 5.5;
```



Basic Java - Array

- Array variables are object variables –
  - When you copy an array variable into another, both variables refer to the same array

```
int[] scores = { 10, 9, 7, 4, 5 };
int[] values = scores;
```




```
scores[3] = 10;
System.out.println(values[3]); // prints 10
```

Basic Java - Array

- Array variables are object variables –
  - Quiz: Consider the following code segment. What is the value of b[2] after the code executes?

```
int[] a = { 0, 1, 2, 3, 4 };
int[] b = { 10, 11, 44, 99 };
a = b;
b = a;
System.out.println('b[2] = ' + b[2])
```

Answer:

```
B[2] = 44;// Draw a diagram may help you grasp

?? What happens to the data block {0,
1, 2, 3, 4}
```

Basic Java - Array

- 2D Array

```
int[][] my2DArray = new int[3][4];
```

  - the number of pairs of square brackets indicates the dimension of the array
  - the first number indicates the row and the second the column

Basic Java - Array

- 2D Array – after initialization

```
int[][] my2DArray = new int[3][4];
```

row \ column	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

Basic Java - Array

- 2D Array – after initialization

```
int[][] my2DArray;
my2DArray = new int[3][];

my2DArray[0] = new int[4];
my2DArray[1] = new int[4];
my2DArray[2] = new int[4];

System.out.println(my2DArray);
System.out.println(my2DArray[1]);
```

Basic Java - Array

- 2D Array – after value assignment

```
int[][] my2DArray = new int[3][4];
my2DArray[2][1] = 8
my2DArray[1][3] = 11
```

row \ column	0	1	2	3
0	0	0	0	0
1	0	0	0	11
2	0	8	0	0

Basic Java - Array

- In Java, 2D Array can be ragged
  - each row does not have to have the same number of columns
    - Can Python do this?
  - Other programming languages do not support this feature

```
int[][] my2DArray;
my2DArray = new int[3][];

my2DArray[0] = new int[2];
my2DArray[1] = new int[3];
my2DArray[2] = new int[4];
```

```
int[][] my2DArray = {{2, 2}, {3,
3, 3}, {4, 4, 4}};
```

Basic Java - Array

- 2D Array – print out the elements
  - To print elements, one by one

```
int[][] my2DArray = {{2, 2}, {3, 3,
3}, {4, 4, 4}};

// 1st row
System.out.println(my2DArray[0][0]);
System.out.println(my2DArray[0][1]);

// 2nd row
System.out.println(my2DArray[1][0]);
System.out.println(my2DArray[1][1]);
System.out.println(my2DArray[1][2]);

…
```

Basic Java

- Array
- Loops and control

Basic Java - loop and control

- Control structure
  - Decision making with if-else statement

```
if(boolean-expression)
      statement;

if(boolean-expression)
{     statement1;
      statement2;
}
else
      statement3;
      //single-statement body needs no { } braces
```

Basic Java - loop and control

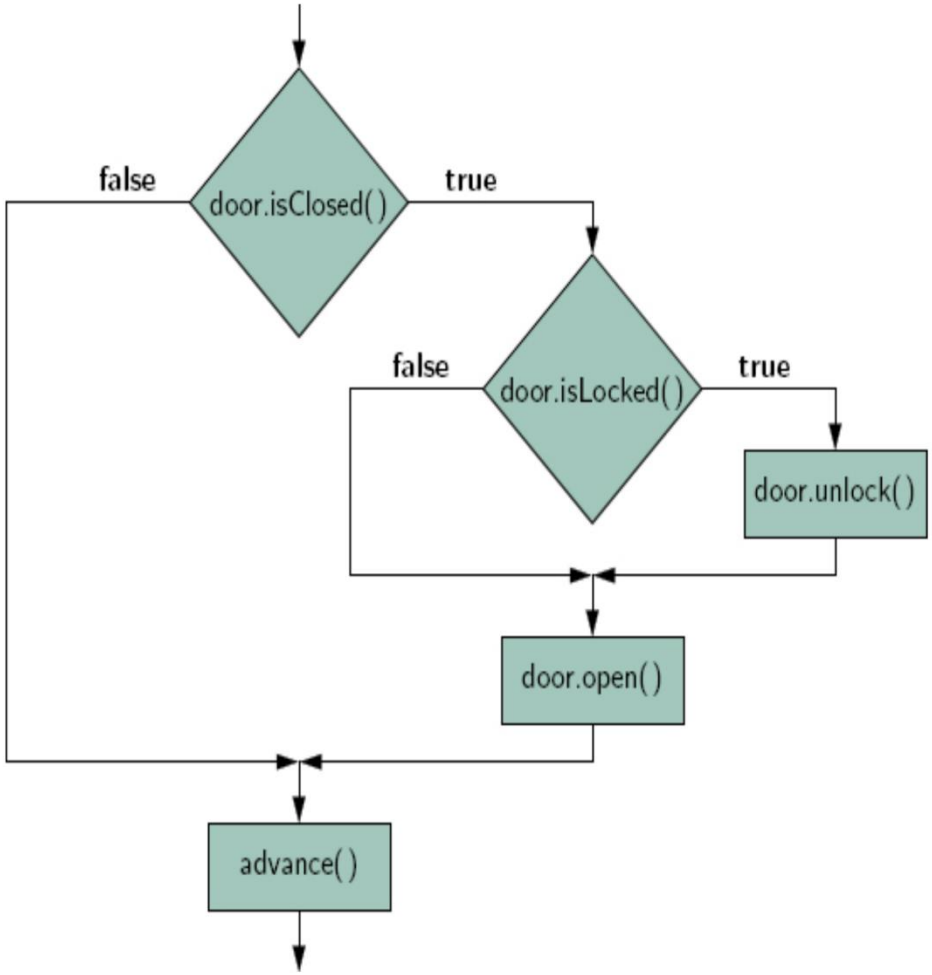
- Control structure - boolean expression: true/false
  - Relational Operators: >, >=, <, <=, ==, !=
  - Logical Operators: &&, ||, !

```
if( x <= X_LIMIT && y <= Y_LIMIT)

if( x!= 10)
```

```
if( x==5 ) { // then do something }

// how about this one?
If (x = 5) { … }
```

Basic Java - loop and control

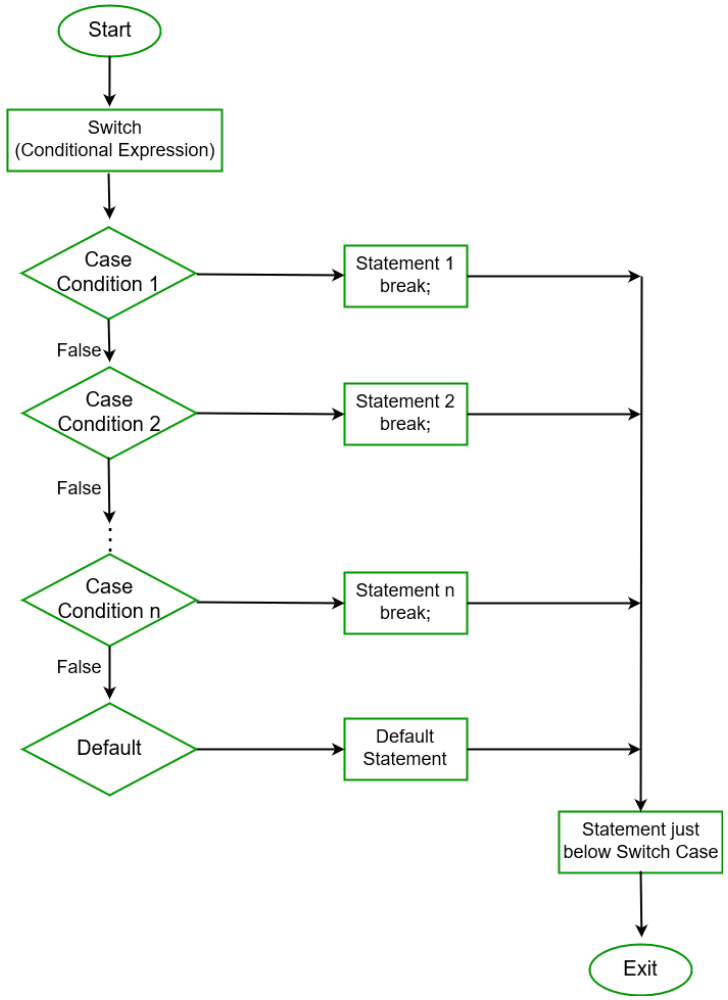
- Control structure - boolean expression: true/false
  - Write codes to illustrate the following:
    - (1) using if-else
    - (2) using if only



Basic Java - loop and control

- Switch statement

```
// switch statement
switch(expression)
{
    // case statements
   // values must be of same type of expression
    case value1 :
        // do something
        break; // break is optional

    case value2 :
       // do something
        break;
        … …

    // used when none of the cases is true.
   // No break is needed in the default case.
    default :
        // do something
}
```



Basic Java - loop and control

- Switch statement

```
switch (DayOfWeek)
{
  case MON:
    System.out.println("This is tough.");
    break;
  case TUE:
    System.out.println("This is getting better.");
    break;
  case WED:
    System.out.println("Half way there.");
    break;
  case THU:
    System.out.println("I can see the light.");
    break;
  case FRI:
    System.out.println("Now we are talking.");
    break;
  default:
    System.out.println("Day off!");
}
```

Basic Java - loop and control

- Switch statement
  - Exercise 1:
    - Write a java program to complete the code of the previous slide
      - You may define an enumerated (enum) type for the day of week
      - What is enum type?

    - An *enumerated or enum type* is a programmer-defined type that is used to restrict a variable to holding one of a fixed set of values defined by the programmer.

```
public enum Day
{
   MON, TUE, WED, THU, FRI, SAT, SUN
};
Day today;
today = Day.TUE;
```

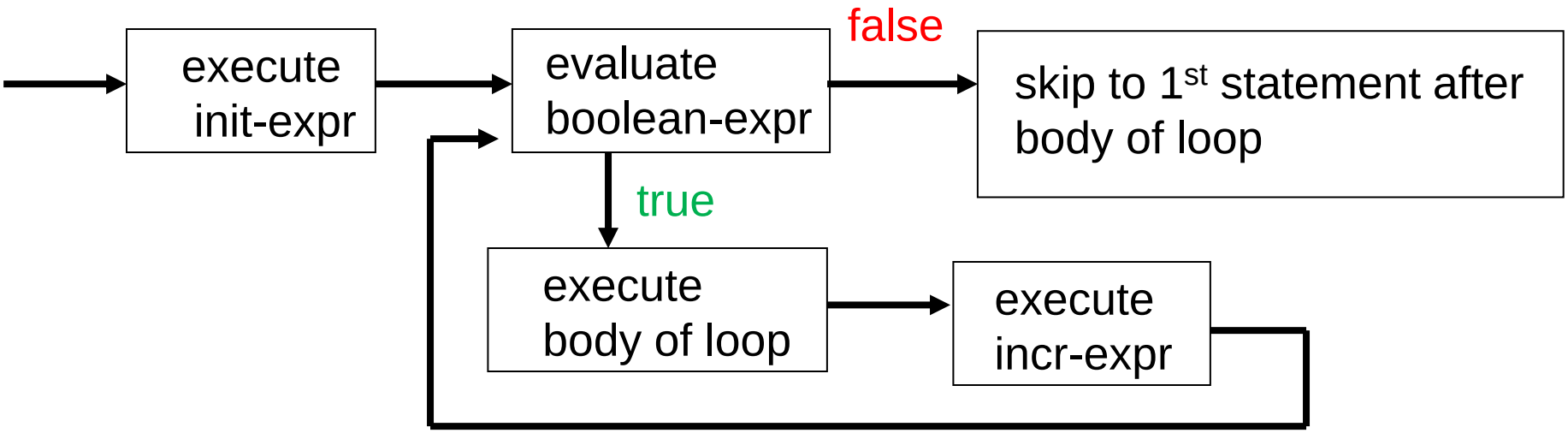
Basic Java – loop and control

- Switch statement
  - Exercise:
    - Write codes that declares an *int* named month whose value represents a month. The code displays the name of the month, based on the value of month, using the switch statement.
    - Could it be done using if-else statement? Which one is neater?

Basic Java - loop and control

- for loops

```
for(init-expr; boolean-expr; incr-expr)
{
      statement1;
}
```



Basic Java - loop and control

- for loops

```
for(init-expr; boolean-expr; incr-expr)
{
      statement1;
}
```

- Example: computes the sum of an array of double values using a for loop:

```
 double total = 0;
 for (int j=0; j < data.length; j++)
 {
   total += data[j];
 }
```

Basic Java - loop and control

- for loops
  - Exercise: complete the following method to find the maximum value within an array of data (one dimension).

```
public double max(double[] data)
{
   double currentMax = data[0];
   … …

}
```

Basic Java - loop and control

- for loops
  - Exercise: complete the following method to find the maximum value within an array of data (two dimension).

```
public double max(double[][] data2D)
{
   double currentMax = data2D[0][0];
   … …
}
```

Basic Java - loop and control

- for loops
  - Since looping through elements of a collection is such a common construct, Java provides a shorthand notation for such loops, called the *for-each loop, or enhanced loop*

```
for (elementType name : container)
            loopBody
```

```
int[] arr = new int[100];
for (int item : arr)
     System.out.println(item);
```

Basic Java - loop and control

- for loops - 1D array print
  - Old style:

```
float data[] = new float[100];
for (int j=0; j < data.length; j++)
    System.out.println (data[j]);
```

  - New way:

```
float data[] = new float[100];
for (float element : data)
    System.out.println (element);
```

Basic Java - loop and control

- for loops
  - How about printing the following 2D arrays?

```
int[][] arr1 = new int[4][5];
int[][] arr2 = {{1, 2, 3, 4 }, { 5, 6, 7,
8}};
```

  - Old style: ??

  - New way: ??

Basic Java - loop and control

- for loops
  - How about printing the following 2D arrays?

```
int[][] arr1 = new int[4][5];
int[][] arr2 = {{1, 2, 3, 4 }, { 5, 6, 7,
8}};
```

  - Old style:

```
for(int i = 0; i<arr2.length; i++)
     for(int j=0; j<arr2[i].length; j++)
          System.out.print(a[i][j] + " ");
```

  - New way:

```
for(int[] i : arr2)
     for (int j : i)
          System.out.print(j + " ");
```

Basic Java - loop and control

- for loops
  - Exercise: Write a program that takes an integer command-line argument N and prints all prime numbers that are less than or equal to N in ascending order

Basic Java - loop and control

- while loops

```
while(boolean-expression)
      statement; // only one statement
```

```
while(boolean-expression)
{
      statement1;
      statement2;
      … …
}
```

Basic Java - loop and control

- while loops

```
while(boolean-expression)
      statement; // only one statement
```

```
while(boolean-expression)
{
      statement1;
      statement2;
      … …
}
```

Basic Java - loop and control

- while loops
  - break or continue

```
while(boolean-expression)
{
      statement1;
      if(…) break; // leave the loop
      … …
}
```

```
while(boolean-expression)
{
      statement1;
      if(…) continue; // to the last statement
inside while
      … …
}
```

Basic Java - loop and control

- while loops
  - break or continue

```
while (true){
   String input = in.next();
   if (input.equals("Q"))break;
   double x = Double.parseDouble(input);
   data.add(x);
}
```

```
while (!done){
    String input = in.next();
    if (input.equals("Q"))  {
        done = true;
        continue; // Jump to the end of the loop body
    }
    double x = Double.parseDouble(input);
    data.add(x);
    // continue statement jumps here
}
```

Basic Java - loop and control

- for loops - break
  - What's the output of the following?

```
double balance = 1000; // money that's not invested
double moneyInvested;  // money that is invested
double moneyReturned;  // money that's earned at end of day
int day;               // current day, ranges from 1 to 90

for (day=1; day<=90; day++)  {
     if (balance < 1 || balance > 5000)
     {
       break;
     }
     balance = moneyInvested = balance / 2.0;
     moneyReturned = moneyInvested * (Math.random() * 2);
     balance += moneyReturned;
 }
 System.out.println("final balance on day "
                    + (day-1) + "is" +  balance);
```

Basic Java - loop and control

- do-while loops

```
do
{
      statement1;
      statement2;
      … …

} while(boolean-expression
```

Basic Java - loop and control

- Difference: while and do-while loop?

```
while(boolean-expression)
      statement; // only one statement
```

```
do
      statement;
while(boolean-expression);
```

Basic Java - loop and control

- while loop:
  - Exercise – what's the output of the following? E.g., Sum = ??

```
i=0;
sum=0;

while(sum<10)
{
   i++;
   sum = sum+1;
}

System.out.println(
     "sum = " + sum);
```

Sum = 10

```
i=0;
sum=0;

while(sum >= 10)
{
   i++;
   sum = sum+1;
}

System.out.println(
     "sum = " + sum);
```

Sum = 0

Basic Java - loop and control

- while loop:
  - Exercise – what's the output of the following?

```
i=0;
sum=0;

while(sum<10)
{
   i++;
   sum = sum-1;
}

System.out.println("i = " + i + " " + "sum = " + sum);
```

Sum = 2147483647
i = ??

Basic Java - loop and control

- What does the following do?

```
int sum = 0;
while(sum < 10);
```

- Empty statement
  - The empty statement is a statement that does nothing.
  - It consists of a semicolon by itself.
  - What's the point of having this empty statement?

```
long bigNum = 100000000;
for(long i=0; i<bigNum; i++);
```

>> a "quick and dirty" way to add a delay to your program, e.g. delay to start a process

Basic Java - loop and control

- Exercise - use the while loop for the following problem
  - You put $10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original investment?
    - You need to declare a few variables

```
int year = 0; balance = 10000; targetBalance = 20000;
rate = 0.05;
while (balance < targetBalance)
{
      year++;
      double interest = balance * rate / 100;
      balance = balance + interest;
}
```

Basic Java - loop and control

- do-while loops
  - Exercise: Insert the missing condition in the following code fragment. The code is intended to compute the sum of a number of integers entered by the user. The loop should stop when the sum exceeds 100.

```
int total = 0;
do
{
   System.out.print("Enter an integer:");
   Scanner in = new Scanner (System.in);
   int value = in.nextInt();
   total = total + value;
}
while ( _____________ );
```

Basic Java - loop and control

- How to copy the contents of an array to another array?
  - There are a few approaches
    - See the Eclipse demo

```
int[] source = …
int[] target …

1: target = source // copy reference only
2: using the loop to copy an element a time

3: target = source.clone();

4: System.arraycopy(…)
5: target = Arrays.copyOf(…);
```