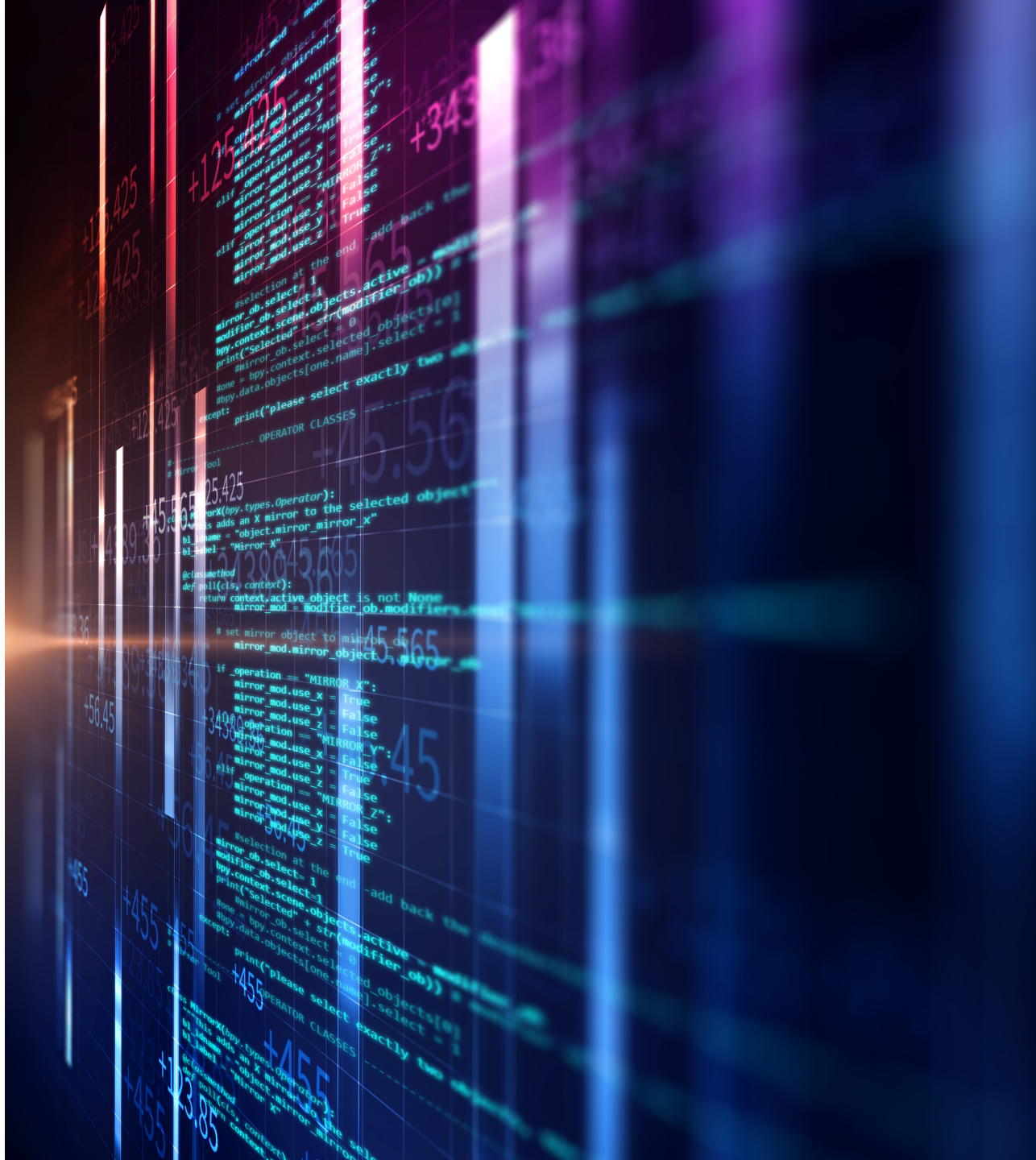
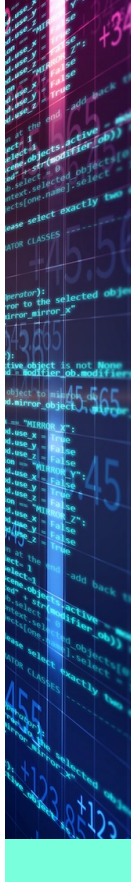
Loops

Brandon Krakowsky

Penn Engineering

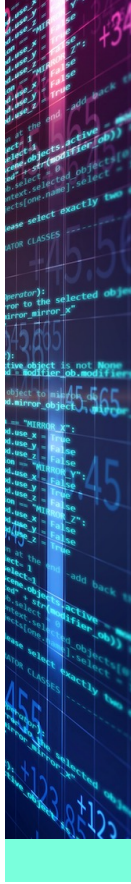


Flow Control: Loops



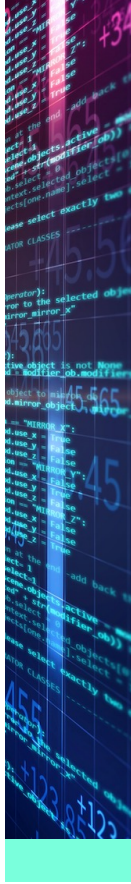
Loops

- Used to repeat a process (block of statements) or perform an operation multiple times
- *for* loops
  - Run a piece of code *for* a given number of times
- *while* loops
  - Run a piece of code indefinitely *while* a condition is met



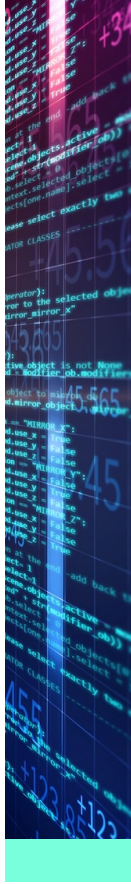
for Loops

- A *for* loop executes code a given number of times



for Loops

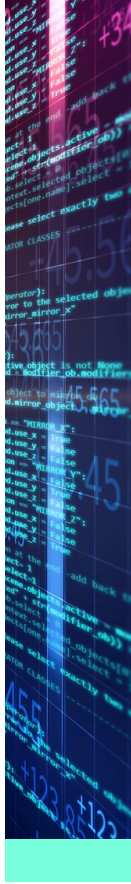
- A *for* loop executes code a given number of times
- A *for* loop has 3 parts:
  - Setting the initial value
  - The condition for entering the loop
  - The change in the loop variable that happens at the end of each loop



for Loops

- A *for* loop executes code a given number of times
- A *for* loop has 3 parts:
  - Setting the initial value
  - The condition for entering the loop
  - The change in the loop variable that happens at the end of each loop
- Simple *for* loop that *iterates* 10 times:

```
for (int i = 0; i < 10; i++) {
    //do stuff here every time loop happens
}
//i is initially set to 0
//i must be less than 10 in order to enter the loop each time
//i is incremented by 1 at the end of each loop (you can't see it)
```

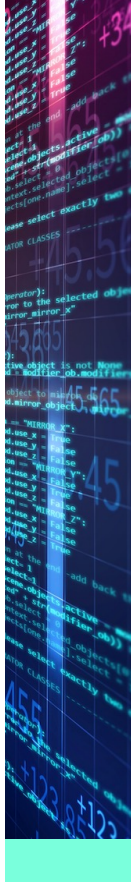


for Loops

- *Iterates* over 10 numbers, from 0 – 9

ForLoops.java

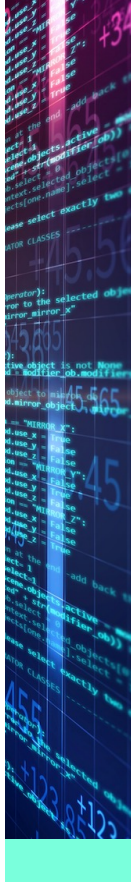
```
1  /**
2   * Introduction to for loops.
3   * @author lbrandon
4   *
5   */
6  public class ForLoops {
7
8      public static void main(String[] args) {
9
10         //iterate 10 times
11         //for each number 0 - 9, print the number
12         for (int i = 0; i < 10; i++) {
13             System.out.println(i);
14         }
15         System.out.println();
16
```



for Loops

- *Iterates* over 10 numbers, from 1 – 10

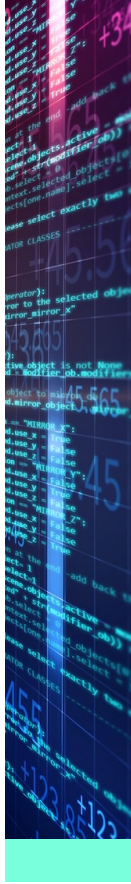
```
17
18          //iterate 10 times
19          //for each number 1 - 10, print the number
20          for (int i = 1; i < 11; i++) {
21              System.out.println(i);
22          }
23          System.out.println();
24
```



for Loops

- Another way! *Iterates* over 10 numbers, printing 1 – 10

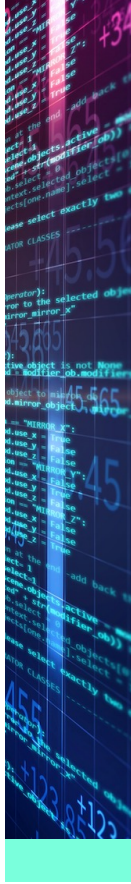
```
24
25
26          //another way
27          //iterate 10 times
28          //for each number 0 – 9, print the number + 1
29          for (int i = 0; i < 10; i++) {
30              System.out.println(i + 1);
31          }
32          System.out.println();
33
```



for Loops

- *Iterates* over 6 numbers, from 1 – 6

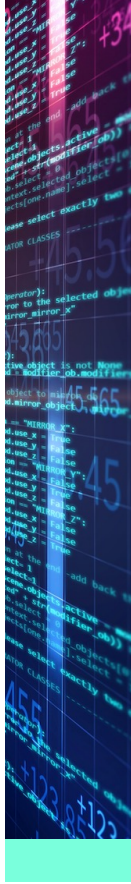
```
34
35          //iterate 6 times
36          //for each number 1 - 6, print the number
37          for (int i = 1; i <= 6; i++) {
38              System.out.println(i);
39          }
40          System.out.println();
41
```



for Loops

- *Iterates* over numbers 0 – 28, skipping every 6 numbers

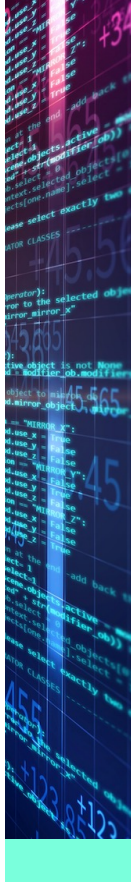
```
42
43          //iterate over numbers 0 – 28
44          //print 0 and multiples of 7
45          for (int i = 0; i <= 28; i++) {
46              if (i % 7 == 0) {
47                  System.out.println(i);
48              }
49          }
50          System.out.println();
51
```



for Loops

- *Iterates* over 6 numbers, counting backwards from 5 – 0

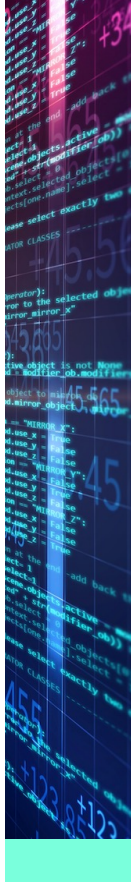
```
51
52          //iterate 6 times (counts backwards)
53          //for each number 5 – 0, print the number
54          for (int i = 5; i >= 0; i--) {
55              System.out.println(i);
56          }
57          System.out.println();
58
```



for Loops

- Here we find the numbers between 1 and 1200 that are odd

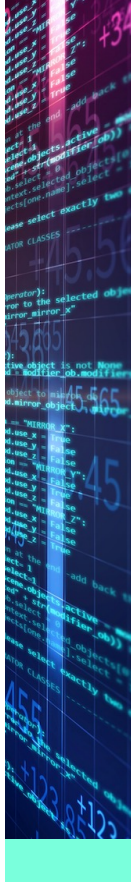
```
62
63          //iterate 1200 times
64          //for each number 1 - 1200, if it's odd, print the number
65          for (int i = 1; i <= 1200; i++) {
66              if (i % 2 != 0) {
67                  System.out.println("i: " + i);
68              }
69          }
70          System.out.println();
71
```



for Loops

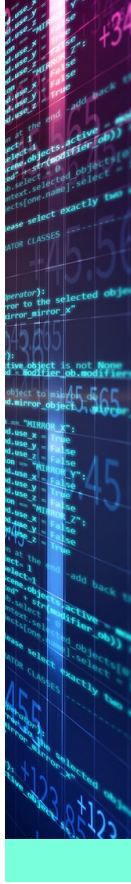
- Here we find the numbers between 1 and 10 that are even
- And we get a count of the even numbers by *incrementing* a count

```
74
75          //initialize a count of even numbers
76          int evenCount = 0;
77
78          //iterate 10 times
79          //for each number 1 - 10, if it's even, print the number and increment the even count
80          for (int i = 1; i <= 10; i++) {
81              if (i % 2 == 0) {
82                  System.out.println("i: " + i);
83                  evenCount++;
84              }
85          }
86
87          //print count of even numbers
88          System.out.println(evenCount + " even number(s)");
89          System.out.println();
90
91      }
92  }
93
```



while Loops

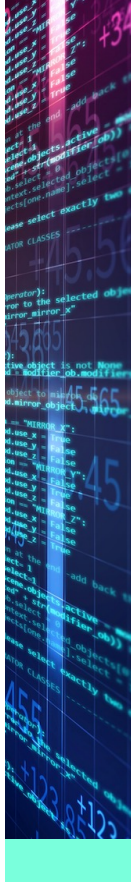
- A *while* loop repeatedly executes code based on a condition
  - Be careful -- if the condition is never met, your loop becomes an *infinite loop* and never stops
  - If this happens, your program could crash!



while Loops

- A *while* loop repeatedly executes code based on a condition
  - Be careful -- if the condition is never met, your loop becomes an *infinite loop* and never stops
  - If this happens, your program could crash!
- Simple *while* loop that *iterates* 10 times:

```
int i = 0;
while (i < 10) {
    //do stuff here every time loop happens
    i++; //manually increment i
}
//i is initially set to 0
//i must be less than 10 in order to enter the loop each time
//code in the loop manually increments i by 1 at the end of each loop
```

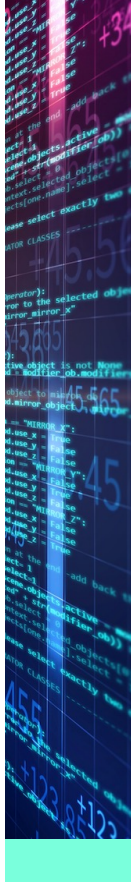


while Loops

- This prints the value of a until it reaches 0

WhileLoops.java

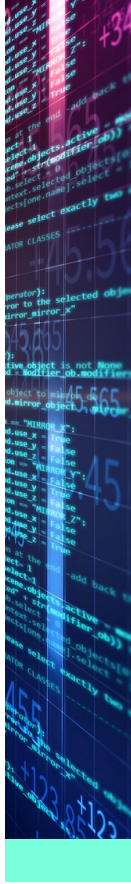
```java
/**
 * Introduction to while loops.
 * @author lbrandon
 *
 */
public class WhileLoops {

    public static void main(String[] args) {

        //prints the value of a until it reaches 0

        int a = 5;

        //while a is greater than 0
        while (a > 0) {
            //enter while loop
            System.out.println("a is being decremented: " + a);
            a -= 1;
        } //exit while loop

        System.out.println();

```



while Loops

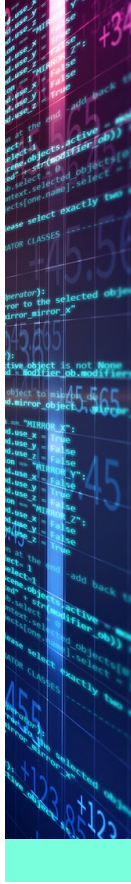
- Here's a program that multiplies x by 2 until an upper limit of 128, starting at 4

```
23
24          //multiplies x by 2 until an upper limit of 128, starting at 4
25
26          int x = 4;
27
28          //while x is less than 128
29          while (x < 128) {
30              //enter while loop
31              x = 2 * x;
32              System.out.println("x is now: " + x);
33          } //exit while loop
34      }
35  }
```



while Loops – Getting User Input

- One use case for a *while* loop is a program that needs to continuously run and “wait” for something to happen, like specific user input

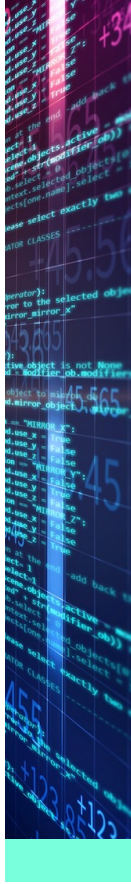


while Loops – Getting User Input

- One use case for a *while* loop is a program that needs to continuously run and “wait” for something to happen, like specific user input
- This program runs until the user says “hello"

WaitForHello.java

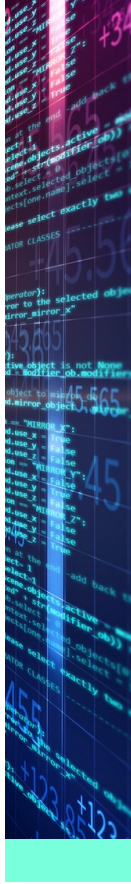
```
1  import java.util.Scanner;
2
3  /**
4   * Waits for input of "hello".
5   * @author lbrandon
6   *
7   */
8  public class WaitForHello {
9
10     public static void main(String[] args) {
11
12         //create a scanner
13         Scanner scan = new Scanner(System.in);
14
15         //prompt for input
16         System.out.println("Hi! Please say hello.");
17         String inp = scan.nextLine();
18
```



while Loops – Getting User Input

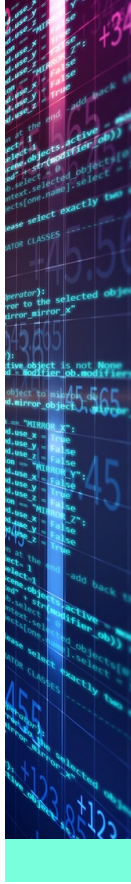
- This program runs until the user says “hello”

```
18
19          //while the input does not equal "hello"
20          //(remember, use the equals method to compare Objects (like Strings))
21          while (!inp.equals("hello")) {
22              //enter while loop and re-prompt for input
23              System.out.println("Please say hello.");
24              inp = scan.nextLine();
25          } //exit while loop
26
27          System.out.println("It's about time!");
28
29          //close the scanner
30          scan.close();
31      }
32  }
33
```



while Loops - Exercise

- Write a program that uses a while loop to test user input of a secret password.
  - If the user inputs "secret", print “Welcome!” and exit the program
  - Otherwise, print “Sorry, the password you entered is incorrect. Please try again.” and prompt the user again

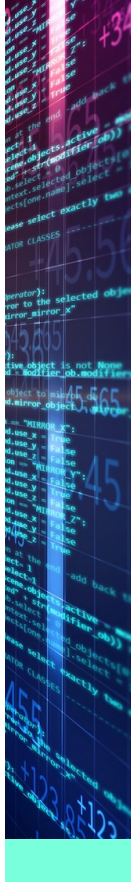


while Loops - Exercise

- Write a program that uses a while loop to test user input of a secret password.
  - If the user inputs "secret", print "Welcome!" and exit the program
  - Otherwise, print "Sorry, the password you entered is incorrect. Please try again." and prompt the user again

TestPassword.java

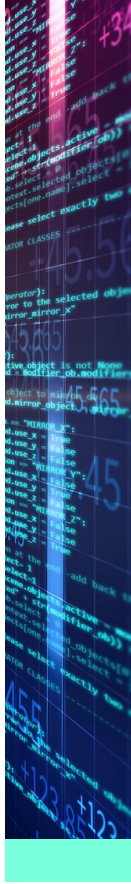
```
1  import java.util.Scanner;
2
3  /**
4   * Waits for input of "secret" password.
5   * @author lbrandon
6   *
7   */
8  public class TestPassword {
9
10     public static void main(String[] args) {
11
12         //create a scanner
13         Scanner scan = new Scanner(System.in);
14
15         String password = "";
16
```



while Loops - Exercise

- Write a program that uses a while loop to test user input of a secret password.

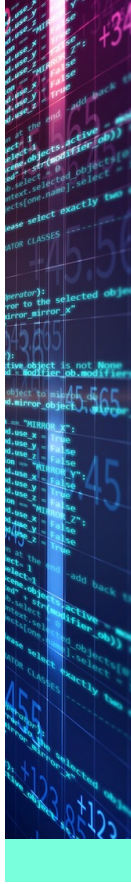
```
16
17          //while the password is not equal to "secret"
18          //(remember, use the equals method to compare Objects (like Strings))
19          while (!password.equals("secret")) {
20              //enter while loop and prompt for input
21              System.out.println("Please enter the password:");
22              password = scan.nextLine();
23
24              //inspect the password and print something accordingly
25              if (password.equals("secret")) {
26                  System.out.println("Welcome!");
27              } else {
28                  System.out.println("Sorry, the password you entered is "
29                          + "incorrect.  Please try again.");
30              }
31          } //exit the while loop
32
```



while Loops - Exercise

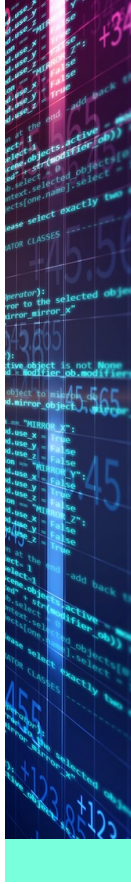
- Write a program that uses a while loop to test user input of a secret password.

```
31
32              //close the scanner
33              scan.close();
34          }
35      }
36
```



Exit a Loop Using *break*

- *break* exits the entire loop immediately

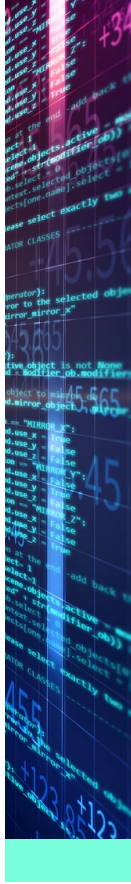


Exit a Loop Using *break*

- *break* exits the entire loop immediately
- This prints 1- 4 only

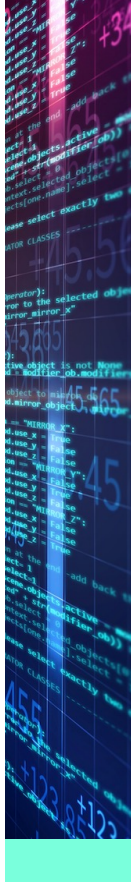
Break.java

```
/**
 * Demonstrates break statement.
 * @author lbrandon
 *
 */
public class Break {

    public static void main(String[] args) {

        int x = 1;

        //while x is less than or equal to 10
        while (x <= 10) {
            //if x is equal to 5
            if (x == 5) {
                break; //this exits the entire while loop!
            }

            //print x
            System.out.println("x is now: " + x);

            //increment x
            x += 1;
        }
    }
}
```



Exit a Loop Using *continue*

- *continue* changes the flow of control and exits the current loop only



Exit a Loop Using *continue*

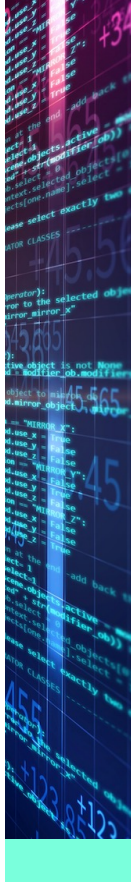
- *continue* changes the flow of control and exits the current loop only
- This prints all the odd numbers between 1 - 20, except those that are multiples of 3

Continue.java

```
/**
 * Demonstrates continue statement.
 * @author lbrandon
 *
 */
public class Continue {

    public static void main(String[] args) {

        //for each number 0 - 20
        for (int u = 0; u <= 20; u++) {

            //if the number is odd
            if (u % 2 != 0) {

                //if the number is a multiple of 3
                if (u % 3 == 0) {
                    continue; //this exits the current iteration of the for loop only
                }

                //print the number
                System.out.println(u);
            }
        }
    }
}

```

Nested Loops

- A *nested loop* is a loop within a loop!
  - For every iteration of the *outer* loop, it runs the *inner* loop

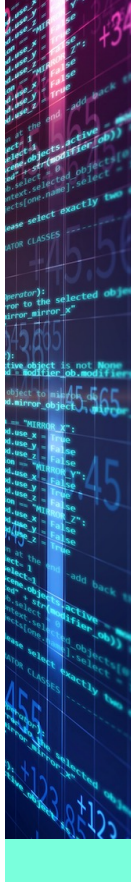


Nested Loops

- A *nested loop* is a loop within a loop!
  - For every iteration of the *outer* loop, it runs the *inner* loop
- What does the following code print?

NestedLoops.java

```
1  /**
2   * Demonstrates nested for loop.
3   * @author lbrandon
4   *
5   */
6  public class NestedLoops {
7
8      public static void main(String[] args) {
9
10         for (int i = 1; i <= 3; i++) {
11             System.out.println("i: " + i); //for each number i
12             for (int j = 1; j <= 2; j++) {
13                 System.out.println("\t j: " + j); //for each number j
14             }
15         }
16         System.out.println();
17
```



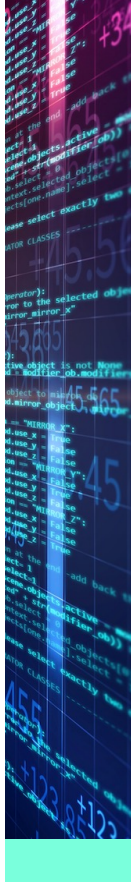
Nested Loops

- A *nested loop* is a loop within a loop!
  - For every iteration of the *outer* loop, it runs the *inner* loop
- What does the following code print?

NestedLoops.java

```
1  /**
2   * Demonstrates nested for loop.
3   * @author lbrandon
4   *
5   */
6  public class NestedLoops {
7
8      public static void main(String[] args) {
9
10         for (int i = 1; i <= 3; i++) {
11             System.out.println("i: " + i); //for each number i
12             for (int j = 1; j <= 2; j++) {
13                 System.out.println("\t j: " + j); //for each number j
14             }
15         }
16         System.out.println();
17
```

```
i: 1
         j: 1
         j: 2
i: 2
         j: 1
         j: 2
i: 3
         j: 1
         j: 2
```



Nested Loops

- A *nested loop* is a loop within a loop!
  - For every iteration of the *outer* loop, it runs the *inner* loop
- What does the following code print?

```
18
19          for (int i = 1; i <= 3; i++) {
20              System.out.println("i: " + i); //for each number i
21              for (int j = 1; j <= 2; j++) {
22                  if (j <= 1) {
23                      //continue to next iteration of current loop
24                      continue;
25                  }
26
27                  System.out.println("\t j: " + j); //for each number j
28              }
29          }
30          System.out.println();
31
```

```
i: 1
        j: 2
i: 2
        j: 2
i: 3
        j: 2
```

Nested Loops

- A *nested loop* is a loop within a loop!
  - For every iteration of the *outer* loop, it runs the *inner* loop
- What does the following code print?

```
33
34          for (int i = 1; i <= 3; i++) {
35              System.out.println("i: " + i); //for each number i
36              for (int j = 1; j <= 2; j++) {
37                  if (j <= 1) {
38                      //break out of current loop only
39                      break;
40                  }
41                  System.out.println("\t j: " + j); //for each number j
42              }
43          }
44          System.out.println();
45      }
46  }
```

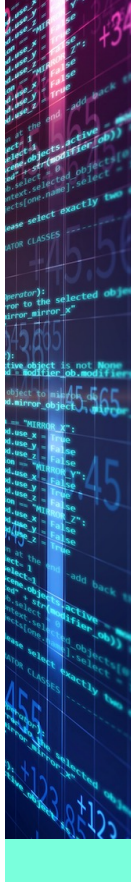
```
i: 1
i: 2
i: 3
```

Nested Loops – Multiplication Table Exercise

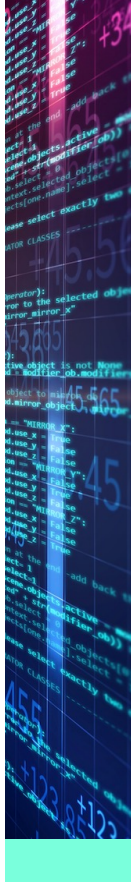
- Let's just make sure we know our multiplication tables

MultiplicationTables.java

```java
/**
 * Multiplication tables.
 * @author lbrandon
 *
 */
public class MultiplicationTables {

    public static void main(String[] args) {

        //for each number i (1 – 10)
        for (int i = 1; i <= 10; i++) {

            //iterate over each number j (1 – 10)
            for (int j = 1; j <= 10; j++) {
                System.out.println(i + " * " + j + " = " + (i * j)); //multiply and print
            }
        }
    }
}
```

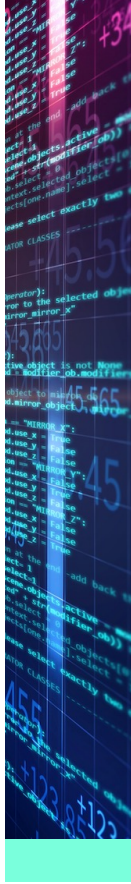


More Programs



Multiplication Table Program

- Write a program that asks the user for a number, then prints out the multiplication table.

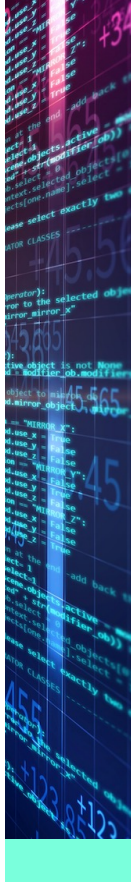


Multiplication Table Program

- Write a program that asks the user for a number, then prints out the multiplication table.

MultiplicationTable.java

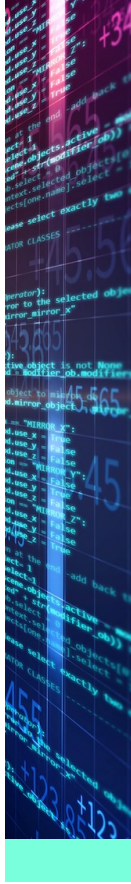
```
1  import java.util.Scanner;
2
3  /**
4   * Prints multiplication table up to 10 for a given number.
5   * @author lbrandon
6   *
7   */
8  public class MultiplicationTable {
9
10     public static void main(String[] args) {
11
12         //create a scanner
13         Scanner scan = new Scanner(System.in);
14
15         //prompt for an int
16         System.out.println("Input an int: ");
17
```



Multiplication Table Program

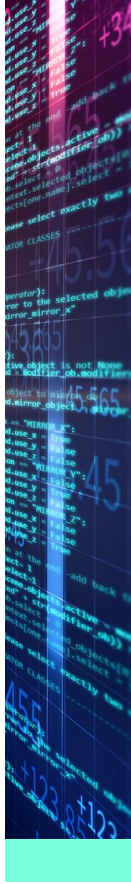
- Write a program that asks the user for a number, then prints out the multiplication table.

```
17
18          //get input as an int
19          int myInt = scan.nextInt();
20          System.out.println("Your number is: " + myInt);
21
22          //print multiplication table up to 10 using myInt
23          for (int t = 1; t < 11; t++) {
24              System.out.println(t + " x " + myInt + " = " + (t * myInt));
25          }
26          System.out.println();
27
28          //close the scanner
29          scan.close();
30
31      }
32  }
```



Calculating Current Balance

- Write a program that calculates what the balance will be after ten years, for a given initial balance and interest rate.

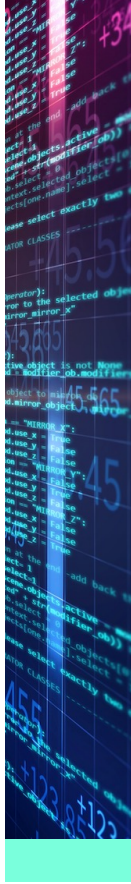


Calculating Current Balance

- Write a program that calculates what the balance will be after ten years, for a given initial balance and interest rate.

CurrentBalance.java

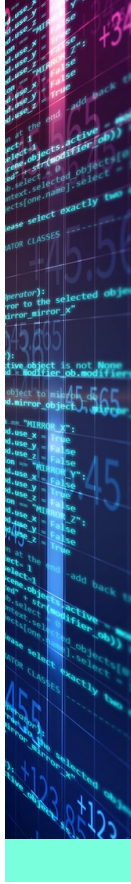
```
1  import java.math.BigDecimal;
3
4  /**
5   * Calculates what the balance will be after ten years,
6   * for a given initial balance and interest rate.
7   * @author lbrandon
8   *
9   */
10 public class CurrentBalance {
11
12     public static void main(String[] args) {
13
14         //initial savings
15         double initialSavings = 10000.0;
16
17         //interest rate
18         double interestRate = 0.05; //5%
19
20         //current savings
21         //defaults to initial savings
22         double currentSavings = initialSavings;
23
```



Calculating Current Balance

- Write a program that calculates what the balance will be after ten years, for a given initial balance and interest rate.

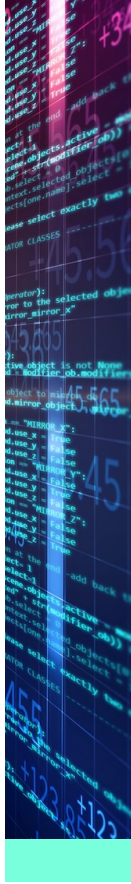
```
23
24          //run 10 times with for loop
25          for (int i = 0; i < 10; i++) {
26              //calculate new current savings
27              currentSavings += currentSavings * interestRate;
28          }
29
30          //print initial savings and current savings
31          System.out.println(initialSavings + " becomes " + currentSavings + " in 10 years");
32
```



Calculating Current Balance

- Write a program that calculates what the balance will be after ten years, for a given initial balance and interest rate.

```
33
34          //another way to do it
35
36          //first, reset the initial values
37          initialSavings = 10000.0;
38          interestRate = 0.05; //5%
39          currentSavings = initialSavings;
40
41          //run 10 times with while loop
42          int i = 0;
43          while (i < 10) {
44              //calculate new current savings
45              currentSavings += currentSavings * interestRate;
46              i++;
47          }
48
49          //print initial savings and current savings
50          System.out.println(initialSavings + " becomes " + currentSavings + " in 10 years");
51
```



Calculating Current Balance

- Write a program that calculates what the balance will be after ten years, for a given initial balance and interest rate.

```
52
53          //how can we round currentSavings?
54
55          //create big decimal with currentSavings
56          BigDecimal bd = new BigDecimal(currentSavings);
57
58          //set config for big decimal
59          bd = bd.setScale(2, RoundingMode.HALF_UP);
60
61          //get rounded currentSavings
62          currentSavings = bd.doubleValue();
63
64          //print initial savings and current savings
65          System.out.println(initialSavings + " becomes " + currentSavings + " in 10 years");
66      }
67  }
```