Repetition Statements

CS 180
Prof. Sunil Prabhakar
Department of Computer Science
Purdue University
Problem

- Write a game program that requires the user to guess a random integer.
- After each input from the user
  - Let the user know if the guess was correct
  - Otherwise, inform the user that the guess was either too high or too low.
- The game ends only when the user correctly guesses the value.
Repetition

To solve this problem, we need the ability to repeat a set of operations (get input, compare with secret and respond) an unknown number of times.

The number is determined by how many guesses the user takes to get it right.

This week we will learn how to repeatedly execute portions of code using **while**, **do-while**, and **for** loops.
public class Guess {
    public static void main(String[] args) {
        int secret, guess;
        boolean done;
        Random random = new Random();
        secret = random.nextInt();
        done = false;
        while (!done) {
            guess = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter your guess."));
            if (guess == secret) {
                done = true;
                System.out.println("You guessed correctly!");
            } else if (guess < secret) {
                System.out.println("Your guess was too low");
            } else {
                System.out.println("Your guess was too high");
            }
        }
    }
}
Control Flow of **while**

1. previous statement;
2. done = false;
3. 
   - !done ?
     - true
       - <get user guess>
       - <print out message>
       - <set done if appropriate>
     - false
   - next statement;
Syntax for the **while** Statement

```java
while ( <boolean expression> )
<statement>
```

Loop body is repeatedly executed as long as boolean expression is **true**.

Example:
```
while (!done) {
    guess = Integer.parseInt(...);
    if (guess == secret) ...
}
```
Example: input check

```java
char grade;
grade = JOptionPane.showInputDialog(null, "Enter grade").charAt(0);
while (grade < 'A' || grade > 'E')
  grade = JOptionPane.showInputDialog(null, "Enter grade").charAt(0);
```

- Only accepts grades 'A' through 'E'
- Note: need for initial input before loop
  - better option **do-while** loop
The do-while Statement

```java
char grade;
do {
    grade = JOptionPane.showInputDialog(null,"Enter grade").charAt(0);
} while ( grade < 'A' || grade > 'E' )
```

- Loop body executed before test (at least once)
- No need for initial input before loop
Common Errors

- Infinite loop
  - if the loop condition never becomes false the loop body will be executed endlessly
  - unless this is desired, ensure that the loop condition will change to false at some point

```java
while(!done){
    guess = . . . ;
    if(guess == secret){
        done = true;
        System.out.println("You guessed correctly!");
    } else if . . .
}
```
Caution: Reals and Equality

Loop 2 terminates, but Loop 1 does not because only an approximation of a real number can be stored in a computer memory.

```
float count = 0.0f;
while ( count != 1.0f ) {
    count = count + 0.33333333f;
}     //seven 3s
```

```
float count = 0.0f;
while ( count != 1.0f ) {
    count = count + 0.33333333f;
}     //eight 3s
```
Loop Pitfall – 2a

1

```java
int result = 0; double cnt = 1.0;
while (cnt <= 10.0){
    cnt += 1.0;
    result++;
}
System.out.println ( result);
```

Using Real Numbers
Loop 1 prints out 10, as expected, but Loop 2 prints out 11. The value 0.1 is not be stored precisely in computer memory.

2

```java
int result = 0; double cnt = 0.0;
while ( cnt <= 1.0){
    cnt += 0.1;
    result++;
}
System.out.println ( result);
```
Control Flow of **do while**

- **previous statement**;
- **grade = JOptionPane(...)**;
- **is grade < 'A' OR grade > 'F' ?**
- **true**;
- **false**;
- **next statement**;
Problem

- Write a program that prints out a multiplication table for a given number input by the user.
  - We will limit our tables to multiples up to 12.
  - The user input should be between 2 and 12.
public class PrintOneTable {
    public static void main(String[] args) {
        int i;
        i = Integer.parseInt(JOptionPane.showInputDialog(null, "Which table would you like?
        System.out.println("1\t\t" + i + "\t=\t" + 1*i);
        System.out.println("2\t\t" + i + "\t=\t" + 2*i);
        System.out.println("3\t\t" + i + "\t=\t" + 3*i);
        . . .
        System.out.println("12\t\t" + i + "\t=\t" + 12*i);
    }
}
Issues

- This is not very convenient.
- What if we wanted to print the table up to multiples of 1000?
  - We would have to add 1000 print statements to our code!
- What if we wanted to change the range of multiples?
- Could use a **while** loop instead.
```java
int i, j;
i = Integer.parseInt(JOptionPane.showInputDialog(null, "Which table would you like?"));  
j = 1;  
while(j < 13) {
    System.out.println(""+j+\t\t" +i+\t\t"+j*i);
    j++;
}
```
This is a very common situation:
- initialize a variable
- repeat a loop body until some condition is true
- update variable in each loop

A `for` loop can be used in this situation.
- makes the three steps explicit
- separate from the loop body

```
for ( <initialization>; <boolean expression>; <increment>  )
    <statement>
```
```java
int i, j;
i = Integer.parseInt(JOptionPane.showInputDialog(null, "Which table would you like?");
for (j=1; j < 13; j++) {
    System.out.println(j + "x" + i + "= " + j*i);
}
```

**Initialize**
- `i`, `j` initialized

**Test**
- Condition `j < 13` tested

**Increment**
- `j` incremented by 1

**Loop body is executed for**
- `j = 1, 2, 3, ..., 12`
Control flow of `for`

```java
for (j = 1; j < 13; j++) {
    System.out.println( ... );
}
```
More for Loop Examples

```cpp
int sum = 0;
for (i = 1; i <= 1000; i++)
    sum += i;
```

Sum of the first 1000 integers

```cpp
int product = 0;
for (i = 2; i <= 100; i += 2)
    product += i;
```

Product of the even numbers through 100

```cpp
for (j = 2; j < 40; j *= 2)
```

```cpp
for (int k = 100; k > 0; k--)
```
Problem

Write a program to print out a multiplication table from 1 through 12.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
<td>30</td>
<td>33</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
<td>40</td>
<td>44</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
<td>54</td>
<td>60</td>
<td>66</td>
<td>72</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
<td>70</td>
<td>77</td>
<td>84</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td>72</td>
<td>80</td>
<td>88</td>
<td>96</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
<td>90</td>
<td>99</td>
<td>108</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>22</td>
<td>33</td>
<td>44</td>
<td>55</td>
<td>66</td>
<td>77</td>
<td>88</td>
<td>99</td>
<td>110</td>
<td>121</td>
<td>132</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>24</td>
<td>36</td>
<td>48</td>
<td>60</td>
<td>72</td>
<td>84</td>
<td>96</td>
<td>108</td>
<td>120</td>
<td>132</td>
<td>144</td>
</tr>
</tbody>
</table>
Generating the Table

```java
int row, col;

for(col=1;col<13;col++)
    System.out.print(" " +col);
System.out.print("\n");

for(row=1; row<13; row++){
    System.out.print(""+row);
    for(col=1;col<13;col++)
        System.out.print(" " + row*col);
    System.out.print("\n");
}
```

Outer loop

Inner loop
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
<td>30</td>
<td>33</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
<td>40</td>
<td>44</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
<td>54</td>
<td>60</td>
<td>66</td>
<td>72</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
<td>70</td>
<td>77</td>
<td>84</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td>72</td>
<td>80</td>
<td>88</td>
<td>96</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
<td>90</td>
<td>99</td>
<td>108</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>22</td>
<td>33</td>
<td>44</td>
<td>55</td>
<td>66</td>
<td>77</td>
<td>88</td>
<td>99</td>
<td>110</td>
<td>121</td>
<td>132</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>24</td>
<td>36</td>
<td>48</td>
<td>60</td>
<td>72</td>
<td>84</td>
<td>96</td>
<td>108</td>
<td>120</td>
<td>132</td>
<td>144</td>
</tr>
</tbody>
</table>
In order to control output, we can use the printf() function.

```java
for(row=1; row<13; row++){
    System.out.printf("%4d", row);
    for(col=1; col<13; col++)
        System.out.printf("%4d", row*col);
    System.out.printf("\n");
}
```

minimum 4 characters gap

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
<td>90</td>
<td>99</td>
<td>108</td>
</tr>
</tbody>
</table>
Formatting Output

- The first argument to printf() is the string to be displayed.
- Each occurrence of a format specifier (e.g., \%4d) is replaced by a matching argument.

```java
int i=10, j=23;
System.out.printf("%3d * %4d = %9d", i, j, i*j);
```
Format Specifiers

- Integer data
  - \%<\text{min width}>d e.g., \%5d

- Real numbers (float and double)
  - \%<\text{min width}>.<\text{decimal places}>f
  - e.g., \%8.5f -- use 5 decimal places

- String
  - \%s or \%10s or \%10.3s \%-10s

- Character
  - \%c or \%3c
The format Method of PrintStream

Instead of using the printf() method of PrintStream, we can achieve the same result by using the format method of PrintStream or a Formatter object.

```
System.out.printf("%6d", 498);
```

is equivalent to

```
Formatter formatter = new Formatter(System.out);
formatter.format("%6d", 498);
```

and equivalent to

```
System.out.format("%6d", 498);
```

See API for details.
Breaking out of a loop

- In some cases, it is necessary to get out of a loop.
- This is achieved using a `break` statement.

```java
for(row=1; row<13; row++){
    System.out.print(""+row);
    for(col=1;col<13;col++){
        System.out.print(" "+row*col);
        if(col>=row)
            break;
    }
    System.out.print("\n");
}
```

```
  1  2  3  4  5  6  7  8 
1  1
2  2  4
3  3  6  9
4  4  8 12 16
5  5 10 15 20 25
6  6 12 18 24 30 36
7  7 14 21 28 35 42 49
8  8 16 24 32 40 48 56 64
9  9 18 27 36 45 54 63 72 81
10 10 20 30 40 50 60 70 80 90
11 11 22 33 44 55 66 77 88 99 110 121
12 12 24 36 48 60 72 84 96 108 120 132 144
```
breaking Out of Outer Loop

```java
for (row=1; row<13; row++) {
    System.out.print("" + row);
    for (col=1; col<13; col++)
        System.out.print(" "+ row*col);
    System.out.print("\n");
    if (col*row>30)
        break;
}
```

```
1 1 2 3 4 5 6 7 8 9 10 11 12
1 1 2 3 4 5 6 7 8 9 10 11 12
2 2 4 6 8 10 12 14 16 18 20 22 24
3 3 6 9 12 15 18 21 24 27 30 33 36
```
### Skipping an iteration

- We can skip the current iteration of a loop using a **continue** statement.
- A continue transfers control to the test statement of the loop.

```java
for (row=1; row<13; row++){
    if (row%2==0)
        continue;
    System.out.printf("%4d", row);
    for (col=1; col<13; col++)
        System.out.printf("%4d", row*col);
    System.out.print("\n");
}
```

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>22</td>
<td>33</td>
<td>44</td>
<td>55</td>
<td>66</td>
<td>77</td>
<td>88</td>
<td>99</td>
</tr>
</tbody>
</table>
Multiple statements in for loop

- The initialization and increment of a for loop can contain multiple statements separated by commas.

```java
int sum;
for (int i=0,sum=0; i<=10; sum+=i,i++) {
}
System.out.println("Sum from 1 to 20 is: " + sum);
```
Confirmation Dialog

- Used to give the user a choice between different buttons.

```java
JOptionPane.showMessageDialog(null, 
    "Are you sure?", 
    "File Deletion Confirmation", 
    JOptionPane.YES_NO_CANCEL_OPTION);
```
Example: Confirmation Dialog

```java
int choice;
choice = JOptionPane.showConfirmDialog(null,
   "Are you sure?",
   "File Deletion Confirmation",
   JOptionPane.YES_NO_CANCEL_OPTION);

if(choice == JOptionPane.YES_OPTION)
   System.out.println("You chose to delete the file");
else if (choice == JOptionPane.NO_OPTION)
   System.out.println("You chose not to delete the file");
else
   System.out.println("You chose to cancel");
```