Self Quiz

Assume \( x=10, z=30 \). Evaluate the following boolean expressions:

\[
\begin{align*}
&x<10 \; || \; x>10 \\
&!((x<20 + z) \; && \; (x+10 \; <= \; 20)) \\
&x==20 \; || \; z==30 \\
&!(x==20) \; || \; (z==30) \\
&!(x==20) \; && \; (z!=20)
\end{align*}
\]

There are two main kinds of conditional statements

- if
- switch

Some people recommend you only use if for style reasons

The switch statement

Syntax (The default case is optional):

\[
\text{switch ( expression )}
\{
\text{case char/byte/short/int constant : statementSequence}
\text{default: statementSequence}
\}
\]

Example

```java
int z=5;
int i=2;
switch ( i )
{
\text{case 1: } z = 1; \text{ break;}
\text{case 2: } z = 2;
\text{case 3: } z = 3; \text{ break;}
\text{default: } z = 0;
}
```

Breaking out

The break statement can occur anywhere within a switch, for, while or do statement and causes execution to break out of the current statement (which may be a loop) and jump to the next statement.

Parameters

Here is the first line of the method declaration:

```java
public boolean equalsIgnoreCase(String anotherString)
```

Here is the way the method is called:

```java
String str1 = "map", str2 = "MAP";
str1.equalsIgnoreCase( str2 );
```

How does the string get into the method???
Pass-by-value

- Basic types are passed into a method by their value
  - This means that the method receives a copy of the variable.

- Can the method change the value of the original variable?
  - No! It can only change the copy.

Pass-by-value for references

- Reference variables are passed by their reference or address
  - This means that the method gets a copy of the address to the object.

- Can the address change within the method?
  - No! Only the copy can change.

Example

```java
public class testPassing {
    int i = 0;
    StringBuffer str = new StringBuffer("rich");

    public static void main( String[] args ) {
        System.out.println("Int: "+i+" String: "+str);
        testVars(i, str);
        System.out.println("Int: "+i+" String: "+str);
    }

    public void testVars( int i, StringBuffer str ) {
        i = 5;
        str = new StringBuffer("poor");
    }
}
```

Self Quiz

- Assume x=10, z=30. Evaluate the following boolean expressions:
  - \( x<10 \, || \, x>10 \)
  - \( ! (x<20 + z) \, \, && \, (x+10 <= 20) \)
- Express the following switch statement using nested if statements:

```java
switch (grade) {
    case 10:
    case 9:
        a = 1;
        b = 2;
        break;
    case 8:
        a = 3;
        b = 4;
        break;
    default:
        a = 5;
        break;
}
```

Intro. to IceBlox

http://www.javaonthebrain.com/brain.html

Game by: Karl Hörnell, April 8 1996

while loop

- Syntax:

```
while ( booleanExpression )
    statement
```
IceBlox Example

```java
while (game != null)
    try
        game.sleep(snooze);
    catch (InterruptedException e) {}
    counter = (counter + 1) % 255;
    switch (gameState)
        case 0: prepareField();
        break;
        case 1: ...
        default: break;
    repaint();
```

Sentinel Controlled Loop

- The loop runs until one or more variable sentinels halt the loop through changing the value of that variable.

Example

```java
int product = 1, number = 1, count = 20, lastNumber;
lastNumber = 2 * count - 1;
while (number <= lastNumber)
    product = product * number;
    number = number + 2;
```

Count-controlled loops

- The loop body is executed for a fixed number of times.

Divide Example

```java
public class Divide {
    public static void main(String args[]) {
        int dividend = 35;
        int divisor = 5;
        int quotient = 0;
        int remainder = dividend;
        while (remainder >= divisor)
            remainder = remainder - divisor;
            quotient = quotient + 1;
        System.out.println( dividend + " / " + divisor + " = " + quotient);
        System.out.println( dividend + " % " + divisor + " = " + remainder);
    }
}
```

do loop

- Syntax:

```
do
    statement
while (booleanExpression);
```
Example

do {
    age = inputBox.getInteger( "Your Age (between 0 and 130): " );
    if ( age < 0 || age > 130 ) {
        messagebox.show( "An invalid age was entered. " +
                         "Please try again." );
    }
} while ( age < 0 || age > 130 );

When to use while?

- do will make sure that you always enter
  the loop at least once
- while may be a better choice if you don’t
  have to enter the loop at least once

Operators

<table>
<thead>
<tr>
<th>Description</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>-=</td>
<td>&gt;=</td>
</tr>
<tr>
<td>*=</td>
<td>&amp;=</td>
</tr>
<tr>
<td>/=</td>
<td>^=</td>
</tr>
<tr>
<td>%=</td>
<td>&gt;&gt;=</td>
</tr>
<tr>
<td>&gt;&gt;=</td>
<td>&lt;&lt;=</td>
</tr>
<tr>
<td>&lt;&lt;=</td>
<td></td>
</tr>
</tbody>
</table>

Review: while loops

while (i<j) // Coins
{
    p=1+(int)(m.random()*blockX);
    q=1+(int)(m.random()*blockY);
    if (playArea[q*(blockX+2)+p]==0)
    {
        if (i<5)
            playArea[q*(blockX+2)+p]=10; // Frozen coin
        else if (i<levIce[effLevel]+5)
            playArea[q*(blockX+2)+p]=2; // Ice cube
        else
            playArea[q*(blockX+2)+p]=1; // Rock
        i++;
    }
}

Decrement and Increment Operators

- Increment x by one: x++
- Decrement x by one: x--

Pre-increment: ++x
Means increment x by 1
before doing the rest of a statement

Post-increment: x++
Means increment x by 1
after doing the rest of a statement

for loops

- Syntax:
  - Each of the expressions is optional, the
    semicolons are not.
  - A for loop is basically a while loop with
    initialization and updating thrown in.

    for( initExpr; booleanExpr; updateExpr )
    
statement
Factorial Example

```java
public class Factorial {
    public static void main(String args[]) {
        int num = 5;
        int fact = 1;
        for (int i = 1; i <= num; i++) {
            fact = fact * i;
        }
        System.out.println(fact);
    }
} // Factorial
```

CharCount Example

```java
public class CharCount {
    public static void main(String args[]) {
        String theString = "Now is the time for all good people ... ";
        char target = 't';
        int count = 0;
        for (int i = 0; i < theString.length(); i++) {
            if (theString.charAt(i) == target) {
                count++;
            }
            System.out.println(target + " appears " + count + " times");
        }
    }
} // CharCount
```

for vs. while

- Use while when you don’t know how many times a loop should run (sentinel controlled loops)
- Use for when you want to run a loop for a fixed number of times (count controlled loops)

Self Test

- Change the following code from for loops into while loops
- What is the value of sum after the nested loop is run?

```java
int sum = 0;
for (int i=0; i<=4; i++)
    for (int j=0; j<=10; j++)
        sum = sum + i;
```