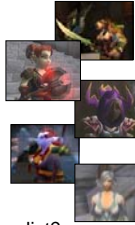


What's This?

- | | |
|---------|------------|
| 1. 3422 | 1. Hunter |
| 2. 8945 | 2. Warrior |
| 3. 5643 | 3. Warlock |
| 4. 3527 | 4. Mage |
| 5. 3472 | 5. Priest |



A list of health values?

A character class list?

Arrays

- o A variable can hold exactly one value
- o Arrays do lists of items
- o Each location in the array is identified by its position/index in the array
 - o Array indices start at 0 and run to one less than the number of locations in the array

1 Dimensional (1-D) Arrays



Arrays

- o What else would it be helpful to make into arrays?
- o Why are the indices helpful?

Arrays

- o Arrays are *objects* but there is no class that array objects are instances of
 - o This means that array variables are reference types
- o Like any reference type, before using an array you must do two things
 1. Declare the identifier (name) that will be used to refer to the array
 2. Create the array (give it space to put values in)
 3. Values inside an array must be of the same type

Arrays

- o Variables of array type are *declared* using bracket ([]) notation:

```
typename[] identifier;  
typename identifier[];
```

Array Creation

- o It is possible to directly initialize/create the values of the array elements using an initializer list:

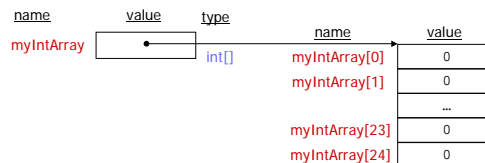
```
int[] n = { 1, 2, 3, 4, 5 };  
float[] m = {1.3f, 2.4f, 3.5f};  
String w[] = {"Tom", "Scary", "Moo Cow"};
```

- o Or the array can be created with the new command:

```
int[] n = new int[5];  
float[] m = new float[5];  
String w[] = new String[3];
```

Creating an Array

```
int[] myIntArray = new int[ 25 ];
```



New and Initialization

- o If the array is created with the new command, it should be initialized to some sort of values

```
int[] n = new int[5];  
  
n[0] = 1;  
n[1] = 2;  
n[2] = 1;  
n[3] = 2;  
n[4] = 3;
```

Create it

- o Create an array of any type. Make it of size 5 and give it values. Print out the values of the array.

Some Answers

```
int rolls[ ] = {20, 91, 5, 88, 21};
```

OR

```
int ages[] = new int[5];  
ages[0] = 20;  
ages[1] = 91;  
ages[2] = 5;  
ages[3] = 88;  
ages[4] = 21;
```

What your program does is similar to:

<http://www.cs.rit.edu/~cs1/lectureNotes/wk6src/IntArray.java>

length

- o An array object has an instance variable, **length**, that gives the number of locations in the array

```
int heads[] = new int[ 10 ];  
System.out.println( heads.length );
```

- o Once created, the length of an array is fixed

Example

```
class DCEAcctGen {
public static void main(String[] args) {
    if (args.length != 4) {
        System.out.println("Please enter first name, " +
            "middle name, last name, and SS# to run " +
            "the program.");
    } else {
        String firstInitial = args[0].substring(0,1);
        String middleInitial = args[1].substring(0,1);
        String lastInitial = args[2].substring(0,1);
        String lastFourSS= args[3].substring(args[3].length()-4, args[3].length());
        System.out.println(firstInitial + middleInitial + lastInitial + lastFourSS );
    }
}
}
```

while loop

o Syntax:

```
while ( booleanExpression )
    statement
```

Useful?

- o The game loop concept is conceptually a while loop
 - o While the game is not ended, continue to update the state machine

IceBlox Example

```
while (game !=null)
{
    try
    {
        game.sleep(snooze);
    } catch (InterruptedException e) {}
    counter=(counter+1)&255;
    switch (gameState)
    {
        case 0: prepareField();
                break;
        <lots of cases inserted here>
        default: break;
    }
    repaint();
}
```

Sentinel Controlled Loop

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

Example

```
int product=1, number=1, count=5, lastNumber;
lastNumber= 2*count -1;
while (number <= lastNumber)
{
    product = product * number;
    number = number + 2;
}
```

Count-controlled loops

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

Divide Example

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

do loop

- o 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?

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

Decrement and Increment Operators

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

Pre-increment: ++x

Post-increment: x++

Means increment x by 1 before doing the rest of a statement

Means increment x by 1 after doing the rest of a statement

for loops

o Syntax:

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

```
for ( initExpr; booleanExpr; updateExpr )  
    statement
```

Factorial Example

```
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
```