Great Video Game Crash of 1983

Napoleon’s Russian Campaign, 1812-1813

The dumpster

Exam 2 Review
Objects, Inheritance, Arrays, Strings

Objects
- **Defined** by template given in as class statement.
- An object is **created** by invoking the class’s constructor using the new keyword.
- An objects is **stored** in a variable declared with class as type
- Values passed to a constructor must be copied to object fields to "stick"

```java
Tree myMaple; // Variable defined as type Tree
void setup() {
    myMaple = new Tree("maple", 30.3); // Create
}

class Tree {
    String name;
    float height;

    Tree(String tname, float theight) {
        name = tname;
        height = theight;
    }

    void draw() {
        fill(0, 255, 0);
        ellipse(random(width), random(height), 50, 50);
    }
}
```
Creating Objects

1. Declare a variable with the class as type
2. Invoke the constructor using the new keyword and assign to variable

```java
Tree myMaple; // Variable defined as type Tree
myMaple = new Tree("maple", 30.3); // Create and assign

// -----
// Two steps combined in one
Tree myMaple = new Tree("maple", 30.3);
```

Using Objects

- variable :: fields (variable inside an object)
- function :: method (function inside an object)

- An variable that stores an object is used to scope access to the fields and methods of that particular object

```java
Tree myMaple;
void setup() {
  myMaple = new Tree("maple", 30.3);
}
void draw() {
  myMaple.draw();
}

class Tree {
  String name;
  float height;
  Tree(String tname, float theight) {
    name = tname;
    height = theight;
  }
  void draw() {
    fill(0, 255, 0);
    rect(10, 10, 50, 300);
  }
}
```

Inheritance

- Superclass (base class) – higher in the hierarchy
- Subclass (child class) – lower in the hierarchy
- A subclass is derived from a superclass
- Subclasses inherit all the fields and methods of their superclass
- Subclasses can override a superclass method by redefining it.
  - They can replace anything by redefining locally
class Person {
    String name; int age;
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    void pr() {
        println(name + " is " + age + " years old.");
    }
    void stats() {}
}

class Student extends Person {
    int year; float GPA;
    Student(String name, int year, float GPA) {
        super(name, 18);
        this.year = year;
        this.GPA = GPA;
    }
    void stats() {
        println(name + " will graduate in " + year);
    }
}

class Employee extends Person {
    float salary; String position; boolean current;
    Employee(String name, int age) {
        super(name, age);
    }
    void hire(String position, float salary) {
        this.position = position;
        this.salary = salary;
        current = true;
    }
    void fire() {
        current = false;
    }
    void stats() {
        if (current) {
            println(name + " works as " + position + " making " + salary);
        } else {
            println(name + " is not working for us.");
        }
    }
}

Arrays - Creating
- A structure that can hold multiple items of a common data type
- Arrays can hold any data type, including objects
- The data type to be held by an array must be declared as part of the array declaration
- Arrays are themselves a kind of type, which is made by adding brackets to the type that the array can hold

Arrays – Creating and Init’ng (3 Steps)
1. Declare an array variable
   - The variable is NOT an array
2. Create an array and assign it to the variable
   - Use the new keyword and size
   - The array is filled with default values
     - int <- 0
     - float <- 0.0
     - boolean <- false;
     - any object including String <- null
3. Fill the array with items of appropriate type

```java
Tree[] trees;

// No array. Only a variable that can hold an array.
```

```java
Tree[] trees;
trees = new Tree[5];

// An empty array. null Tree objects.
```
```java
Tree[] trees;
trees = new Tree[5];
trees[0] = new Tree("maple", 20.0);
trees[1] = new Tree("oak", 203.4);

An array with two Tree objects.
```

```java
Tree[] trees;
trees = new Tree[5];
for (int i=0; i<5; i++) {
    trees[i] = new Tree("maple"+i, random(200.0));
}

An array with five Tree objects.
```

```java
int[] ages;
ages = new int[5];
for (int i=0; i<5; i++) {
    ages[i] = 10 + 2*i;
}

An array with five integers.
```

```java
int[] ages = new int[5];
// Same as
// int[] ages;
// ages = new int[5];
An empty array. Default ints (0).
```
Arrays – Using

- An item in an array is accessed by following an array variable with square brackets containing the item number (index)
- Array indexes start with 0
- Once accessed with brackets, the result can be used as if it was the item at the location in the array

Arrays of arrays (2D Arrays)

- If an array can be made of any type by adding brackets, and ...
- an array is a kind of type, then ...
- an array of arrays should be possible by adding a second set of brackets

```java
boolean[][] cell2;  // A variable that holds an array of boolean arrays
```

- Two-dimensional array
... an array of arrays
Proving a 2D array is an array of arrays

- Access fields and methods of top-level array

```java
void setup() {
    boolean[][] cell2;
    cell2 = new boolean[5][5];  // Create array of arrays
    println( cell2[0].length );  // Access array
    println( cell2[1][2] );      // Access array in array
}
```

Making Strings

- Declaring String objects with no chars

```java
String myName;
String myName = new String();
```

- Declaring String objects init’d w/ char array

```java
String myName = "Fred";
String myName = new String("Fred");
```

### String class methods

- `charAt(index)`
  - Returns the character at the specified index
- `equals(anotherString)`
  - Compares a string to a specified object
- `equalsIgnoreCase(anotherString)`
  - S/A ignoring case (i.e. 'A' == 'a')
- `indexOf(char)`
  - Returns the index value of the first occurrence of a character within the input string
- `length()`
  - Returns the number of characters in the input string
- `substring(startIndex, endIndex)`
  - Returns a new string that is part of the input string
- `toLowerCase()`
  - Converts all the characters to lower case
- `toUpperCase()`
  - Converts all the characters to upper case
- `concat(anotherString)`
  - Concatenates String with anotherString

### Built-in String functions (not methods)

- `split(bigString, splitChar)`
  - Breaks a String into a String Array, splitting on splitChar
  - Returns new String Array
- `splitTokens(bigString, splitCharString)`
  - Breaks a String into a String Array, splitting on any char in splitCharString
- `join(stringArray, joinChar)`
  - Builds a new String by concatenating all Strings in stringArray, placing joinChar between each
- `inverseOf(split) function` 
- `text(theString, x, y)`
  - `text(theString, x, y, width, height)`
  - Draws theString on the sketch at (x, y)
Given the commands:
```java
String aPalindrome = "a man, a plan, a canal Panama";
String[] strs = splitTokens(aPalindrome, ",");
```
Answer the following questions:

(3 pts) What will be the length of `strs`?

a) 1  

b) 2  

c) 3  

d) 4  

(3 pts) What will be the value of `strs[1]`?

a) "a man"  

b) "a plan"  

c) "a canal Panama"  

d) 3  

(3 pts) Write the expression used to obtain the number of elements in `strs`.  

The following program was designed to count and print the number of duplicates in the `myArray` String array. Unfortunately, it doesn’t work properly. When I test it with the given data, it tells me that I have 11 duplicates, but I know that there are only two. Fix the program so that it works correctly.

```java
// Count and print the number of duplicate strings in myArray
void setup() {
  int count = 0;
  for (int i=0; i<myArray.length; i++) {
    for (int j=0; j<myArray.length; j++) {
      if (myArray[i].equals(myArray[j])) {
        count++;
      }
    }
    println("There are "+ count+" duplicates.");
  }
}
```