**Week 5 ITMD 411**

**• Quiz 1 ( Tests and Quizzes in Blackboard ) by end of this week ( online )**

**• Sample Quiz 1 (BB Tests and Quizzes )**

**• Gaddis Textbook Files and Blackboard**

**• Graded Work Returned**

**• TA: Wendy Crump ( [wcrump@hawk.iit.edu](mailto:wcrump@hawk.iit.edu) )**

**• The Arc ( for Tutoring in Herman Hall )** [**http://www.iit.edu/arc**](http://www.iit.edu/arc)

**• Grading Rubric: Professionalism / Input and Output Specifications**

**• Lab Assignment Lab 3 ( Caesar ) due end of this week**

**• Lab Assignment Lab 4 ( Stock Transactions ) due end of next week**

**• Reading Assignment**

**Starting Out With Java: From Control Structures through Objects**

**by**

**Tony Gaddis**

[**http://www.barnesandnoble.com**](http://www.barnesandnoble.com)

**Chapter 6 A First Look at Classes**

**6.1 Classes and Objects**

**6.2 Instance Fields and Methods**

**6.3 Constructors**

**6.4 Overloading Methods and Constructors**

**6.5 Scope of Instance Fields**

**6.6 Packages and import Statements**

**6.7 Focus on Object-Oriented Design: Finding the Classes and Their**

**Responsibilities**

**6.8 Common Errors to Avoid**

**Chapter 9 A Second Look at Classes and Objects**

**9.1 Static Class Members**

**9.2 Passing Objects As Arguments to Methods**

**9.3 Returning Objects from Methods**

**9.4 The toString Method**

**9.5 Writing an equals Method**

**9.6 Methods That Copy Objects**

**9.7 Aggregation**

**9.8 The this Reference Variable**

**9.9 Enumerated Types**

**9.10 Garbage Collection**

**9.11 Focus on Object-Oriented Design: Class Collaboration**

**9.12 Common Errors to Avoid**

**Chapter 11 Inheritance**

**11.1 What Is Inheritance?**

**11.2 Calling the Superclass Constructor**

**11.3 Overriding Superclass Methods**

**11.4 Protected Members**

**11.5 Chains of Inheritance**

**11.6 The Object Class**

**11.7 Polymorphism**

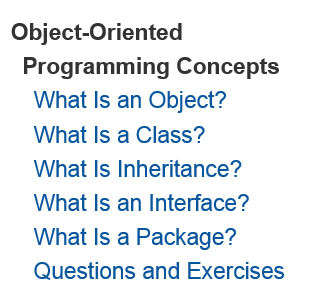
**11.8 Abstract Classes and Abstract Methods**

**11.9 Interfaces**

**11.10 Common Errors to Avoid**

**• Java Classes**

[**http://docs.oracle.com/javase/tutorial/java/concepts/index.html**](http://docs.oracle.com/javase/tutorial/java/concepts/index.html)



**• Key Phrases and Concepts**

**1.Real-world objects contain state and behavior.**

**2.A software object's state is stored in fields.**

**3.A software object's behavior is exposed through methods.**

**4.Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data encapsulation.**

**5.A blueprint for a software object is called a class.**

**6.Common behavior can be defined in a superclass and inherited into a subclass using the extends keyword.**

**7.A collection of methods with no implementation is called an interface.**

**8.A namespace that organizes classes and interfaces by functionality is called a package.**

**9.The term API stands for Application Programming Interface.**

**• Questions**

**Consider the following class:**

**public class IdentifyMyParts {**



**public static int x = 7;**

**public int y = 3;**



**}**

**Question: What are the class variables?**

**Answer: x**

**Question: What are the instance variables?**

**Answer: y**

**Question: What is the output from the following code:**

**IdentifyMyParts a = new IdentifyMyParts();**

**IdentifyMyParts b = new IdentifyMyParts();**

**a.y = 5;**



**b.y = 6;**



**a.x = 1;**



**b.x = 2;**



**System.out.println("a.y = " + a.y);**



**System.out.println("b.y = " + b.y);**

**System.out.println("a.x = " + a.x);**

**System.out.println("b.x = " + b.x);**

**System.out.println("IdentifyMyParts.x = " + IdentifyMyParts.x);**

**Answer: Here is the output:**

**a.y = 5**



**b.y = 6**



**a.x = 2**



**b.x = 2**



**IdentifyMyParts.x = 2**

**Because x is defined as a public static int in the class IdentifyMyParts, every reference to x will have the value that was last assigned because x is a static variable (and therefore a class variable) shared across all instances of the class. That is, there is only one x: when the value of x changes in any instance it affects the value of x for all instances of IdentifyMyParts.**

**static example: the company name**



**instance variable: the employee id**

**Question: What's wrong with the following program?**

**public class SomethingIsWrong {**

**public static void main(String[] args) {**

**Rectangle myRect;**

**myRect.width = 40;**

**myRect.height = 50;**

**System.out.println("myRect's area is " + myRect.area());**

**}**

**}**

**Answer: The code never creates a Rectangle object. With this simple program, the compiler generates an error. However, in a more realistic situation, myRect might be initialized to null in one place, say in a constructor, and used later. In that case, the program will compile just fine, but will generate a NullPointerException at runtime.**

**2.Question: The following code creates one array and one string object. How many references to those objects exist after the code executes? Is either object eligible for garbage collection?**

**...**

**String[] students = new String[10];**

**String studentName = "Peter Smith";**

**students[0] = studentName;**

**studentName = null;**

**...**

**Answer: There is one reference to the students array and that array has one reference to the string Peter Smith. Neither object is eligible for garbage collection.**

**3.Question: How does a program destroy an object that it creates?**

**Answer: A program does not explicitly destroy objects. A program can set all references to an object to null so that it becomes eligible for garbage collection. But the program does not actually destroy objects.**

**Exercises**

**1.Exercise: Fix the program called SomethingIsWrong shown in Question 1.**

**Answer: See SomethingIsRight:**

**public class SomethingIsRight {**

**public static void main(String[] args) {**

**Rectangle myRect = new Rectangle();**

**myRect.width = 40;**

**myRect.height = 50;**

**System.out.println("myRect's area is " + myRect.area());**

**}**

**}**

**2.Exercise: Given the following class, called NumberHolder, write some code that creates an instance of the class, initializes its two member variables, and then displays the value of each member variable.**

**public class NumberHolder {**

**public int anInt;**

**public float aFloat;**

**}**

**Answer: See NumberHolderDisplay:**

**public class NumberHolderDisplay {**

**public static void main(String[] args) {**

**NumberHolder aNumberHolder = new NumberHolder();**

**aNumberHolder.anInt = 1;**

**aNumberHolder.aFloat = 2.3f;**

**System.out.println(aNumberHolder.anInt);**

**System.out.println(aNumberHolder.aFloat);**

**}**

**}**

**• Java Inheritance**

[**http://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html**](http://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html)

[**http://www.javabeginner.com/learn-java/java-inheritance**](http://www.javabeginner.com/learn-java/java-inheritance)

**• Data Structures - Java Syntax ( static Variables and Methods )**

**static variables implies:**

**-Who are we?**

**or**

**-What is assigned to us?**

**package myJ;**

**public class Ex1 {**

**static int *Var1*=77; //Static integer variable**

**String Var2;//non-static string variable**

**public static void main(String args[])**

**{**

**Ex1 ob1 = new Ex1();**

**Ex1 ob2 = new Ex1();**

**ob1.*Var1*=88;**

**ob1.Var2="I'm Object1";**

**ob2.Var2="I'm Object2";**

**System.*out*.println("ob1 integer:"+ob1.*Var1*);**

**System.*out*.println("ob1 String:"+ob1.Var2);**

**System.*out*.println("ob2 integer:"+ob2.*Var1*);**

**System.*out*.println("ob2 String:"+ob2.Var2);**

**System.*out*.println("static variable Var1 Access from Ex2   
 class:"+Ex2.*Var1*);**

**}**

**}**

**class Ex2 {**

**static int *Var1*=11; //Static integer variable**

**String Var2;//non-static string variable**

**}**

**/\***

**Output:**

**ob1 integer:88**

**ob1 String:I'm Object1**

**ob2 integer:88**

**ob2 String:I'm Object2**

**static variable Var1 Access from Ex2 class:11**

**In above example String variable is non-static and integer variable is Static.**

**So you can see that String variable value is different for both objects but integer variable value is common for both the instances as all the objects share**

**the same copy of a static variable.**

**\*/**

**package myJ;**

**class Ex3 {**

**public static void main(String args[])**

**{**

**Ex1 ob1 = new Ex1();**

**System.*out*.println("static variable Access from Ex1 class inside  
 Ex2 class:"+Ex2.*Var1*);**

**}**

**}**

**/\***

**static variable Access from Ex1 class inside Ex2 class:11**

**\*/**

**• Data Structures - Our Queue Example**

**• Data Structures - Our Stack Example**

**//Stack.java**

**import java.util.Arrays;**

**public class Stack**

**{**

**private int count;**

**private string data;**

**public Stack()**

**{**

**data = new String[8];**

**count = 0;**

**}**

**void expandCapacity()**

**{**

**data = Arrays.copyOf(data, data.length \* 2);**

**}**

**void push(string e)**

**{**

**if (count == data.length)**

**expandCapacity();**

**data[count++] = e;**

**}**

**string pop() throws Exception**

**{**

**if (count <= 0)**

**{**

**throw new Exception("stack empty");**

**}**

**count--;**

**return data[count];**

**}**

**boolean isEmpty()**

**{**

**return count == 0;**

**}**

**int size()**

**{**

**return count;**

**}**

**public static void main(String[] args) throws Exception**

**{**

**Stack<String> s = new Stack<String>();**

**s.push("Alice");**

**s.push("Bob");**

**s.push(“Jagdeep”);**

**s.push(“Anwar”);**

**while (!s.isEmpty())**

**System.out.println(s.pop());**

**}**

**}**

**• Data Structures – Cryptographic Systems**

Vigenère cipher

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |

Note that each row of the table corresponds to a Caesar Cipher. The first row is a shift of 0; the second is a shift of 1; and the last is a shift of 25.

The Vigenere cipher uses this table together with a keyword to encipher a message. For example, suppose we wish to encipher the plaintext message:

TO BE OR NOT TO BE THAT IS THE QUESTION

using the keyword RELATIONS. We begin by writing the keyword, repeated as many times as necessary, above the plaintext message. To derive the ciphertext using the tableau, for each letter in the plaintext, one finds the intersection of the row given by the corresponding keyword letter and the column given by the plaintext letter itself to pick out the ciphertext letter.

Keyword: RELAT IONS R ELATI ONS RE LATIO NS REL

Plaintext: TOBEO RNOTT OBETH ATIST HEQUE STION

Ciphertext: KSMEH ZBBLK SMEMP OGAJX SEJCS FLZSY

Decipherment of an encrypted message is equally straightforward. One writes the keyword repeatedly above the message:

Keyword: RELAT IONSR ELATI ONSRE LATIO NSREL

Ciphertext: KSMEH ZBBLK SMEMP OGAJX SEJCS FLZSY

Plaintext: TOBEO RNOTT OBETH ATIST HEQUE STION

This time one uses the keyword letter to pick a row of the table and then traces across the row to get the column containing the ciphertext letter. The index of that column is the plaintext letter.

**• [ Useful Links - Java ]**

**Java Online Compiler**

[**http://www.jdoodle.com/**](http://www.jdoodle.com/)

**Java Quiz**

[**http://www.javatpoint.com/core-java-quiz**](http://www.javatpoint.com/core-java-quiz)

**Java Interview Questions**

[**http://www.javatpoint.com/corejava-interview-questions**](http://www.javatpoint.com/corejava-interview-questions)

**Static Variables**

[**http://www.roseindia.net/java/beginners/staticvariable.shtml**](http://www.roseindia.net/java/beginners/staticvariable.shtml)

[**http://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html**](http://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html)

[**http://stackoverflow.com/questions/3699010/java-static-variables**](http://stackoverflow.com/questions/3699010/java-static-variables)

[**http://www.javatpoint.com/static-keyword-in-java**](http://www.javatpoint.com/static-keyword-in-java)

**Java Quiz**

[**http://www.javatpoint.com/core-java-quiz**](http://www.javatpoint.com/core-java-quiz)

**Java Interview Questions**

[**http://www.javatpoint.com/corejava-interview-questions**](http://www.javatpoint.com/corejava-interview-questions)