3) What is the name of the type of method which is called when an object is instantiated?

**Constructor.**

9) T/F Public methods of a subclass are visible within the methods of a superclass.

**False**

13) Match the statements to the corresponding terms:

|  |  |  |
| --- | --- | --- |
| **I** | **abstract** | 1. **What occurs when a variable in a smaller scope is given the same name as another active variable in a larger scope** |
| **F** | **adapter** | 1. **Occurs when a subclass method is dynamically executed during run-time via a superclass reference.** |
| **E** | **default constructor** | 1. **What a class does to an interface to gain a “is-a” relationship with it** |
| **H** | **final** | 1. **A variable type which points to an instantiated object** |
| **J** | **generic** | 1. **A method provided for free if a class doesn’t define its own methods** |
| **C** | **implements** | 1. **A class with empty implementations of all of an interface’s methods** |
| **K** | **JVM** | 1. **A Java keyword which requires a section of code or method to gain an object’s lock before executing (or, a java keyword useful for making thread-safe code)** |
| **B** | **polymorphism** | 1. **Variable type specifier useful for assigning constant values** |
| **D** | **reference** | 1. **A class which cannot be instantiated** |
| **A** | **shadowing** | 1. **A method which is parameterized by one (or more) type parameters** |
| **G** | **synchronized** | 1. **A program which executes bytecode** |

14. For the following three sentences, inset the terms public, protected, and private where they belong, possibly including multiple terms on each line:

a. **protected, public**: an instance variable with this specifier may be directly accessed from within a subclass.

b. **public**: every .java file must contain a class with this specifier.

c. **private, protected**: instance variables with this specifier cannot be accessed directly from unrelated classes.

**15.** Complete the code for SomeClass so that it has a class variable named “count” which keeps track of how many objects of this class have been instantiated. Also, define the public method getCount so that it can be used to get that count, without having to use a reference to a SomeClass object.

public class SomeClass {

**private** **static int count;**

**public SomeClass(){**

**count++;**

**}**

**public static int getCount(){**

**return count;**

**}**

**}**

…. (following is some code from another class which used SomeClass) ….

// Finish the code one line of code that causes the value of count to be printed to the console

// without using the keyword “new”

\_System.out.println(SomeClass.getCount());

16. Given the definition of the class MyClass, finish the first line of code to instantiate a MyClass object, and pass 10 to its constructor. Then, finish the second line of code to retrieve the value of its myInt instance variable. Finally, tell what the value of myInt will be at that point. Watch for shadowing!

class MyClass {

**Write your code here:**

**MyClass mc = new MyClass (10);**

**int myInt = mc. getMyInt ();**

**what is the value of myInt if the above was correctly done?**

**0**

protected int myInt;

public MyClass(int inInt){

int myInt = this.myInt;

myInt = inInt;

}

public int getMyInt () {return myInt;

}

}

17. Write a method named “matches” which takes 2 parameters, each an array of integers. It should return the total count of how many times an element of the first occurs within the second. For example, if the first array is {1,2,3,3,10,20} and the second is {20,3,200,99,20}, then it should return 4.

public Class Test

{

public static void main (String [] args)

{

Int []test1= {1,2,3,3,10,20};

int []test1= {20,3,200,99,20};

System.out.println(matches (test1, test2));

}

public static int matches (int [] a, int [] b)

{

int count = 0;

for (int i = 0; i< a. length; i++)

{

for (int j = 0; j < b. length; j++)

{

if (a[i]==b[j])

{

count++;

}

}

}

return count;

}

}

18. Given the following partial class definition for Pinball, implement a method which will cause the given output from these two lines of code:

Pinball p = new Pinball (“Medieval Madness”, 1977);

System.out.println(p);

Output: “The pinball machine Medieval Madness was produced in 1997.”

class Pinball {

private String title;

private int year;

public Pinball (String titleIn, int yearIn) {

title = titleIn;

year = yearIn;

}

}

// Implement your method here, and make sure it is declared so that the compiler will verify

// that it is replacing an inherited version of the method you define

@Override

public String toString () {

return String. Format ("The pinball machine %s was produced in %d.", title, year);

}

}

19) The method “getElement” below has the potential to throw a “java.lang.ArrayIndexOutOfBoundsException” if an index beyond the bounds of the array is passed to it. Add the necessary code to “getElement” to use java’s exception handling mechanism to print “Out of Bounds!” to the console if that occurs and then simply return “0”.

**public static int getElement (int [] array, int index) {**

**try {**

**return array[index];**

**}**

**catch (ArrayIndexOutOfBoundsException e) {**

**System.out.println("Out of bounds!");**

**return 0;**

**}**

**}**

20) Given the following definition for a class named Student, provide code to define a subclass named Undergraduate. It should include the following:

Undergraduate (subclass of Student):

1. An Instance variable which is public and a String, and is used to store the major.
2. A class variable which is public and an integer, and is used to store the number of CS majors.
3. A constructor with three arguments which are String specifying the first name, last name, and major (in that order). The constructor should cause all instance variables to be correctly set (including ensuring that gradeLevel is set to ”undergrad”) and it should also make sure that the number of CS majors is correctly updated if the major is “CS”.
4. An overridden version of the method “getRecord” which adds the major to the record which is returned.

class Student {

private String firstName;

private String lastName;

private String gradeLevel;

public Student(String first, String last, String level) {

firstName = first;

lastName = last;

gradeLevel = level;

}

public ArrayList<String> getRecord() {

ArrayList<String> record = new ArrayList<String> ();

record.add ("First: " + firstName);

record.add ("Last: " + lastName);

record.add ("Grade: " + gradeLevel);

return record;

}

}

**public class Undergraduate extends Student {**

**public String major;**

**public static int num CS=0;**

**public Undergraduate (String firstName, String lastName, String major)**

**{**

**super (firstName, lastName, "undergrad");**

**this.major = major;**

**if (major == "CS")**

**return numCS++;**

**}**

**@override**

**public ArrayList<String>getRecord ()**

**{**

**ArrayList<String>record= super. getRecord ();**

**record.add ("major: "+ major);**

**return record;**

**}**

**}**

21. Assuming that the Undergraduate class is correctly implemented, finish implementing the following method which takes as a parameter an ArrayList of references to Student objects. It should calculate and return a floating point value which is, of the total number of students who are undergraduates majoring in CS, the percentage of those who are included in the ArrayList.

**public static float percentOfCSTotal (ArrayList<Student> students) {**

**int undergraduate = 0;**

**int numCS= 0;**

**for (int i=0; i < students. Size (); i++) {**

**if(students.get(i) instanceof Undergraduate) {**

**undergraduate++;**

**if (major == "CS") {**

**numCS ++;**

**}**

**}**

**}**

**float csTotal = (numCS / undergraduate) ;**

**return csTotal;**

**}**

22. Define a generic method which takes as parameters (1) a reference to an object of a generic type and (2) an array of references to that generic type. Your method must loop through the array, and for the first reference in the array which is null, assign that reference to point to the same object as the reference parameter, then return. The method should have void as its return type.

**Public <T> void genericFunction (T object, T [] arr) {**

**for (int i = 0; i < arr.length; i++ ){**

**if(arr[i] == null) {**

**arr[i] = object;**

**return;**

**}**

**}**

**}**

23. Given the following code which creates a from object of type MyFrame, add the code necessary to have the frame prosess mouse event using the class MyMouseListener, and implement MyMouseListener so that it causes the string “Button 1 pressed” to be printed to the console whenever button 1 is **pressed.** [refer to documentation provided for the MouseListener interface, and also note that the MouseEvent class has a method “public int getButton ()” which return the number associated with the mouse button which was pushed.

import javax.swing.JFrame;

import java.awt.event. MouseListener;

import java.awt.event. MouseEvent;

public class EventTest {

public static void main (String [] args) {

MyFrame frame = new MyFrame ();

frame.setVisible(true);

}

class MyFrame extends JFrame {

public MyFrame () {

setSize (200,200);

setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);

addMouseListener (new MyMouseListener ());

}

}

Class MyMouseListener **implements MouseListener** {

**public void mousePressed (MouseEvent e) {**

**if (e. getButton () ==1) {**

**System.out.println(“Button 1 pressed”);**

**}**

**}**

**public void mouseReleased (MouseEvent e) { }**

**public void mouseEntered (MouseEvent e) { }**

**public void mouseExited (MouseEvent e) { }**

**public void mouseClicked (MouseEvent e) { }**

**}**

}

24. Assume that the following is part of a java application allows group of users to form chat groups. The class ChatGroup keeps track of the text that should currently be displayed to the group. It has method with the following signatures implemented:

public void printAllText ()

public void addMessage (String msg)

public void removeMessage (String msg)

However, the class ChatGroup is not thread-safe. A single ChatGroup object is created for each group, but if multiple users attempt to call those method simultaneously, the data structure which stores the messages can become corrupted. Therefore, you must complete the implementation of a new class called ThreadSafeChatGroup which is designed so that a new instance of ThreadSafeChatGroup can be created for each user of a chat group (which all share the same instance of ChatGroup, and that is assumed to be passed into the constructor).

You must make the necessary additions to ThreadSafeChatGroup so that it has three methods with the exact same signatures as those of ChatGroup, and those method in ThreadSafeChatGroup call the corresponding methods of the ChatGroup object, while also ensuring thread-safe across all ThreadSafeChatGroup object using the same ChatGroup object.

**public class ThreadSafeChatGroup {**

**private ChatGroup group;**

**public TreadSafeChatGroup (ChatGroup groupln) {**

**group = groupln;**

**}**

**public void PrintAllText () {**

**synchronized(group) {**

**group.PrintAllText();**

**}**

**}**

**public void addMessage (String msg) {**

**synchronized(group) {**

**group.addMessage (msg);**

**}**

**}**

**public void removeMessage (String msg) {**

**synchronized(group) {**

**group.removeMessage (msg);**

**}**

**}**

**}**