1) (15 points) Write the output for each piece of code. If the code gives an error, write “ERROR”.

Show your work and the changes in memory.

|  |  |  |
| --- | --- | --- |
|  | Code | Output |
| a | q1\_trace(11);public static void q1\_trace(int x){ int a = x/2; while(a > 0){ if(a % 3 == 0){ System.out.println("FUN a="+a); } else if(a % 2 == 0){ System.out.println("BORING a="+a); } else{ System.out.println("YIKES"); } a--;} |  |
| b | q2\_trace();public static void q2\_trace(){ for(int i=3;i<10; i=i+2){ int j=i+1; System.out.println(i\*j); }} |  |
| c | String[] arrA ={"candy", "cookies","chocolate"};String[] arrB ={"broccoli", "beans", "barley"};arrA[0]= "banana"; arrB = arrA;arrB[2]="jello";arrA[1]="corn";System.out.println(“A Contents:”);for(int i=0; i<arrA.length; i++){ System.out.println(arrA[i]);}System.out.println(“\nB Contents:”);for(int i=0; i<arrB.length; i++){ System.out.println(arrB[i]);} |  |

2) The Java class Pet is started below.

This class has **four** private instance variables:

* name, which is a String representing the name of the pet
* age, which is an int representing the pet’s age in whole years.
* isVaccinated, which is a boolean, representing whether the pet has its shots or not.
* tricks,which is a String Array,containing all the tricks the pet has learned.

public class Pet {

 private String name;

 private int age;

 private String[] tricks;

 private boolean isVaccinated;

//complete the class as per the instructions below

a) (5 points) Write the default constructor which assigns the following values to the variables

|  |  |
| --- | --- |
| **Instance Variable** | **Value** |
| name | “Buddy” |
| age | 0 |
| tricks | new String [5]  |
| isVaccinated | false |

b) (6 points) Write a modified constructor that takes in 3 arguments and assigns the values to the matching private instance variables name, age, isVaccinated of the Pet Object. Set the array of tricks’s size to 5.

c) (8 points) Write the getter(accessor) and setter(modifier) methods for 2 of the 4 variables: name, isVaccinated.

Assume the others have been written for you.

|  |  |
| --- | --- |
|  |  |
|  |  |

d) (8 points) Write a **private** helper method **getTricksAsString()**

It should **return a single String** containing all the **values** in the tricks array **separated by commas**.

Contents of String [ ] of tricks: 

String returned by getter: “sit, beg, roll, jump, hide”

e) (8 points) Write a **private** helper method **getVaccinationStatusAsString()**.

It should **return a String** of either **“has vaccinations”** or **“not vaccinated”** depending on the true/false value of the isVaccinated variable.

f) (8 points) Fill in the toString method so that it returns a well formatted String consisting of the names and values of all 4 of a Pet’s instance variables.

Use the helper methods you wrote in parts 2d) and 2e) to complete the toString method

Example:

Pet Details: name=Buddy, age=10, has vaccinations, tricks= sit, beg, roll, jump, hide

Pet Details: name=Fluffy, age=1, not vaccinated, tricks= dance, stand, fetch

@Override

public String toString(){

}

g) (8 points) Fill in the equals method so that it returns true if the values of age,name,isVaccinated match for the calling object and the passed in object. Return false otherwise.

@Override

public boolean equals(Object o){

}

h) (7 points) Use your constructor from part (2b) to create 3 separate instances of class Pet with the following values:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **pet1** | **values** |  | **pet2** | **values** |  | **pet32** | **values** |
| name | “Buddy” |  | name | “Fluffy” |  | name | “Tiger” |
| age | 10 |  | age | 1 |  | age | 19 |
| tricks | ----- |  | tricks | ----- |  | tricks | ----- |
| isVaccinated | true |  | isVaccinated | false |  | isVaccinated | true |

3) Use the Interface BasicActions and the partial classes Bird and Tiger to complete the code.

1. (7 points) For both classes, fill in the **missing code** in the classes **Tiger** and **Bird** so that they become a child/subclass of **Pet** and implement the **BasicActions** Interface.
2. (7 points) For both classes, fill in the default constructor so that it calls the super class’s default constructor and assigns **true** to the isVaccinated variable.
3. (7 points) Fill in the body of the **eat** method so that it uses the getter method and/or variables when appropriate to print a customized sentence.

Example:

 The eat method would print the following:

 A Bird named “Polly” whose favoriteFood is “seeds”: **“Chirp Chirp I am Polly I like seeds.”**

 A Tiger named “Tremendo” whose favoriteFood is “meat”: **“Roar! I am Tremendo I like meat.”**

1. (7 points) Fill in the body of the **sleep** method so that it prints out the following:

For the **Bird** class, **“chirp on a pirch”**

For the **Tiger** class, **“zzRoarZZ on the ground”**.

|  |
| --- |
| public interface BasicActions { public void eat(); public void sleep();} |
| public class Bird\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ { private String favoriteFood; public Bird(){  } @Override public void eat() {  } @Override public void sleep() {    }} | public class Tiger\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ { private String favoriteFood; public Tiger(){  } @Override public void eat(){} @Override public void sleep() {     }} |

4) (9 points) Write a block of code that interacts with the user via the console using the Scanner class.

Ask the user to enter the current month as the number representing the month.

Determine what the expected weather is for that month based on the rules in the table below.

Tell the user the expected weather in a nicely formatted sentence as shown in the example below.

|  |  |
| --- | --- |
| **MONTH** | **WEATHER** |
| Jan, Feb, March, December  | COLD |
| April, May, September, October, November  | MILD |
| June,July,August  | HOT |

Example:

Output: Enter the number for the current month:

Input: 10

Output: In October the weather is mild.

This page is intentionally left blank.