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

Show your work and the changes in memory. (5pts each subquestion a-c)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Code | |  |
| a | int [] numsA = new int[4];  int [] numsB = {7,12,9,10};  for(int i=0; i<numsA.length; i++){  if(i % 2 == 0){  numsA[i] = numsB[i];  }  else{  numsA[i] = i+3;  }  } | | Array Content numsA  Array Content numsB |
| b | char[] arrA={'c','a','d','b','u','r','y'};  char[]arrB={'c','h','o','c','o','l','a','t','e','s'};  char[] arrC={'c','a','n','d','i','e','s'};  arrA[2] = arrB[7];  arrA[6] = arrB[9];  arrC[4] = arrB[5];  arrB = arrA;  //arrA[1] = arrB[5];  arrA[3] = 'f'; | | Array Content arrA  Array Content arrB  Array Content arrC |
| c | public static void q1C(char c, int x){  switch(c){  case 'z':  System.out.println("zoo");  int b = x;  while(b > 0){  System.out.println(b+" b");  b--;  }  break;  case 'm':  System.out.println("monkeys");  case 'b':  for(int i=0; i<x; i++){  System.out.println("bananas");  }  break;  default:  System.out.println("chaos");  break;  }  } | Method Call | Output |
| q1C(‘z’, 3); |  |
| q1C(‘m’, 2); |  |
| q1C(‘b’, 3); |  |
| q1C(‘a’, 3); |  |

2) The Java class **Player** is described below. Complete the Player class as per the instructions.

This class has **4** private instance variables: (5pts each subquestion a-h)

* playerName, which is a String representing the name of the Player object
* jerseyNum, which is an int representing the jersey number
* isAllowedToPlay,which is a boolean representing whether the Player can play or not
* positionsPlayed, which is a String [] containing the positions the Player plays

2a) Declare the 4 private instance variables, but do not assign any values to them.

2b) Write the default constructor which assigns the following default values to the variables

|  |  |
| --- | --- |
| **Instance Variable** | **Value** |
| playerName | “Doe Player” |
| jerseyNum | -1 |
| isAllowedToPlay | true |
| positionsPlayed | 5 |

2c) Write an overloaded constructor that takes in 4 arguments and assigns the values to the matching private instance variables playerName, jerseyNum, isAllowedToPlay, positionsPlayed of the Player Object.

2d) Write the **getter** and **setter** methods for **2** of the variables: **playerName**,and **jerseyNum**.

You can Assume the other getters and setters have been written for you.

2e) Write a method **getPositionsAsString()**

It should **return a single String** containing all the **values** in the positionsPlayed array **separated by commas**. Be sure there is no trailing comma at the end.

Example contents of String array **positionsPlayed**:(assuming a player was capable of playing ALL positions well)



String would be “Center, Small Forward, Power Forward, Shooting Guard, Point Guard”

2f) Override the **toString()** method so that it returns a well formatted String consisting of the names and values of all of the Players instance variables. Use the helper method from 2e.

Example Strings for three different players:

“Player: name=Antetokounmpo, jersey num=34, is allowed to play, positions played = small forward, shooting guard, point guard”

“Player: name=Jordan, jersey num=23, is not allowed to play, positions played = small forward, shooting guard, point guard”

“Player: name=Bird, jersey num=33, is not allowed to play, positions played = small forward, power forward”

2g) Override the **equals(Object o)** method so that it returns **true** if the values of playerName, jerseyNum, isAllowedToPlay,match for the calling object and the passed in object. Return **false** otherwise.

Note: don’t worry about the positionsPlayed for this question.

2h) Use your constructor from part (2c) to create 3 separate instances of class Player with the following values:

“Player: name=Antetokounmpo, jersey num=34, is allowed to play, positions played = small forward, shooting guard, point guard”

“Player: name=Jordan, jersey num=23, is not allowed to play, positions played = small forward, shooting guard, point guard”

“Player: name=Bird, jersey num=33, is not allowed to play, positions played = small forward, power forward”

//Write your answer to question 2 here

3) Use the Interface **Skills** shown below, and the code from question 2 to complete question 3.

|  |  |  |
| --- | --- | --- |
| public interface Skills {  public void move();  public void shoot();  public void pass(String otherPlayerName);  } | | |

1. Create the 2 classes, **MostValuablePlayer** and **BenchedPlayer** so that they become a child/subclass of **Player** and implement the **Skills** Interface.
2. For both classes, create the default constructor so that it calls the super class’s default constructor and assigns the default value shown below for the child class’s boolean hasSkills instance variable.

**MostValuablePlayer** assign **true** to the hasSkills variable.

**BenchedPlayer** assign **false** to the hasSkills variable.

1. Override the **move()** method using getters when necessary:

**MostValuablePlayer** print: “So fast! You can’t see me”

**BenchedPlayer** print: “Oh so slow...lost the ball”

1. Override the **shoot()** method using getters when necessary:

**MostValuablePlayer** print: “SWISH! “

**BenchedPlayer** print: “Missed… no skills”

1. Override the **pass(String otherPlayerName)** method using getters when necessary:

**MostValuablePlayer** print: “I am passing to “+otherPlayerName

**BenchedPlayer** print: “I tried to pass to ”+otherPlayerName

|  |  |
| --- | --- |
| //Answer question 3 here BenchedPlayer | //Answer question 3 here MostValuablePlayer |

//More room to write your answer if needed

4) Remember the Recursive Search algorithm Binary Search. Write a **recursive** method named binarySearch that will do the following:

* Take in an **array** of type **char**, a **target** of type **char**, the **startIndex,** and the **endIndex**.
* Use recursion to find a match for the target and return the index location of the match.
* Return -1 if no match exists in the array.

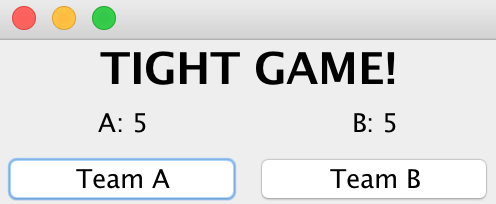
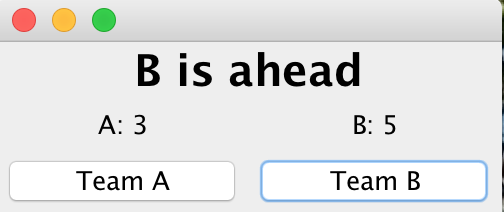
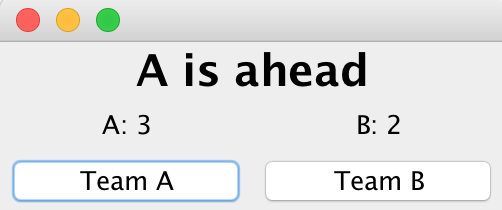
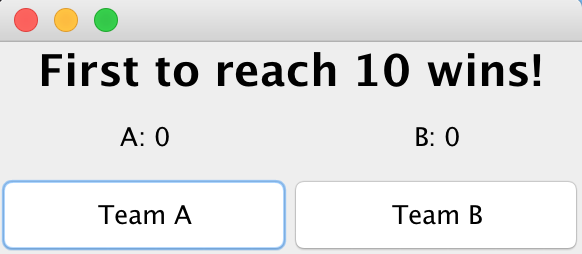
5) The **Java GUI Project ScoreTracker** is started below.

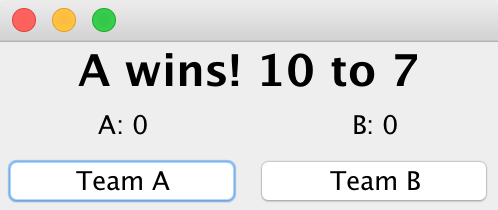
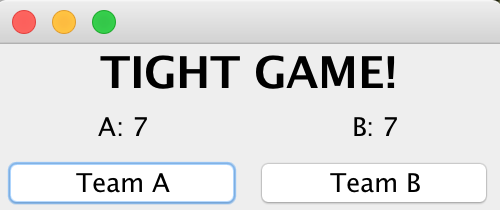
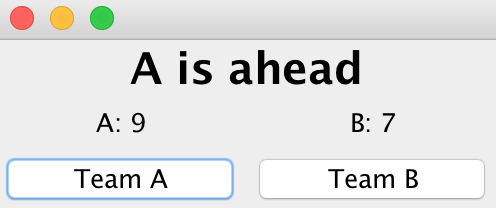
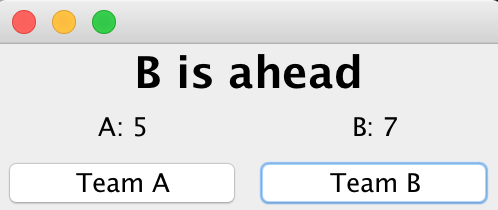
Do the following to complete the **GameScoreJPanel**. (Don’t worry about the JFrame… it’s code is not here)

* 1. Fill in everything that is missing from the class to assure the following:
     1. The class is a JPanel
     2. The class implements the ActionListener interface

1. Create new instances of the JButtons so they say “Team A” , “Team B” accordingly.
   * 1. Make the JButtons listen for actions.
     2. Add them both to the nested jpScore JPanel
2. Create the method **updateWinnerLabel()** such that it uses scoreA and scoreB to do the following:
   * 1. Compare scoreA to scoreB, and set the jlblWinner text
     2. If noone has won yet (score is less than MAX\_POINTS) do the following:
        + Set the text to show which team is ahead
        + Set the text to show “Tight game” when the score is tied
     3. If one of the teams wins (the score equals MAX\_POINTS) do the following:
        + Set the text of the jlblWinner to show the winner, then call **resetScores()**
3. Create the method **resetScores** so that it will do the following
   * 1. Set the variables scoreA and scoreB values’s to 0
     2. Set the text on jlblA and jlblB so they read “A: 0” and “B: 0” using the scores variables
4. Create the method **actionPerformed** below, such that it does the following:
   * 1. Determines which button was clicked
     2. Increments the appropriate team’s score
     3. Sets the text on jlblA or jlblB to display the updated score
     4. Calls updateWinnerLabel() so it can update the jlWinner status accordingly

Below are some screenshots of the game’s progress.. Not all screenshots, just some.





public class GameScoreJPanel\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_{//5a your code here

private JPanel jpScore;

private JLabel jlblA, jlblB, jlblWinner;

private JButton jbA, jbB;

private int scoreA = 0; private int scoreB = 0;

public static final int MAX\_POINTS = 10;

public GameScoreJPanel(){

setLayout(new BorderLayout());

jpScore = new JPanel();

jpScore.setLayout(new GridLayout(2,2));

jlblA = new JLabel("A: "+ scoreA);

jlblB = new JLabel("B: "+ scoreB); //5b Write your code below

jpScore.add(jlblA);

jpScore.add(jlblB);

add(jpScore, BorderLayout.CENTER);

add(jlblWinner, BorderLayout.NORTH);

} //end of GameScoreJPanel Constructor

//5c Write code for the updateWinnerLabel method below

//5d Write code for the resetScores() method below

//5e Fill in the actionPerformed method below

public void actionPerformed(ActionEvent e) {

}

}