1. **(15pts) Trace the code and write the output**

|  |  |  |
| --- | --- | --- |
|  | **Code** | **Output / Memory Content** |
| **a** | int [] numsA = new int[5];  int [] numsB = new int[5];  for(int i=0; i<numsA.length; i++){  numsA[i] = (2 \* i);  numsB[i] = i;  if(i % 2 == 1){  numsA[i] = (3 \* i);  numsB[i] = i;  }  } | numsA Content |
| numsB Content |
| **b** | char [] arrA={'g','i','r','r','a','f','f','e'};  char [] arrB={'g','o','r','r','i','l','l','a'};  arrA[3] = arrB[5];  arrB[2] = arrA[6];  arrB = arrA;  arrA[7]=arrB[4]; | arrA |
| arrB |
| **c** | **Circle all the bugs/errors in the code shown below. (Logical and Syntactical) Hint: There are 10** | |
| private static int countA, countB, countC, countD, countF, countInvalid =0;  public static boolean getLetterGradeFromNumBAD(int score){  if(score => 90){ countA++; return 'A'; }  else if(score >= 60){ countB++; return 'B';  else if(score >= 70){ countC--; return 'C'; }  else if(score >= 80){ countD++; return 'D' }  if else(score == 0){ countF++; return 'F'; }  else countInvalid++; return 1 ; }  } | |

1. **(15pts)** The **Java GUI Project FindMaxMin** is started below.

Do the following to complete the **FindMaxMinJPanel**.

* 1. Fill in everything that is missing from the class to assure the following:
     1. Create the instances of the JButtons so they say “max” , “min” accordingly.
     2. Make the **max** and the **min** JButtons listen for actions.
     3. Add them both to the btnPanel
  2. Fill in the body of the method **getDoubleFromTextField** such that it gets the text from the JTextField object that was passed in and returns a double. The method should catch any exceptions and throw a new Exception with an appropriate message.
  3. Fill in the body of the method **actionPerformed** below, such that it does the following:
     1. Takes the values from jtf1 and jtf2 using the method **getDoubleFromTextField** from part b
     2. Identifies the action command “max” or “min”.
     3. Finds which number is greater or less
     4. Displays a sentence on the labelOut stating which number is max or min.
     5. Include any Exception Handling as needed.

|  |
| --- |
| public class FindMaxMinJPanel extends JPanel implements ActionListener{  private JLabel lblOut;  private JTextField jtf1, jtf2;  private JButton jbMax, jbMin;  public FindMaxMinJPanel(){  lblOut = new JLabel("Find Max or Min of 2 nums");  jtf1 = new JTextField("num1", 8);  jtf2 = new JTextField("num2", 8);  JPanel jtfPanel = new JPanel();  jtfPanel.setLayout(new GridLayout(1,2));  jtfPanel.add(jtf1);  jtfPanel.add(jtf2);  add(lblOut);  add(jtfPanel);//has jtf1 and jtf2 in it  JPanel btnPanel = new JPanel();  btnPanel.setLayout(new GridLayout(1,2));  **//part a - put your code here**  add(btnPanel);//has all btns in it at this point  }  **//part b - put your code here - create and fill in the method** **getDoubleFromTextField**  @Override  public void actionPerformed(ActionEvent e) {  **//part c - put your code here - fill in the method**  String actCmd =  }  } |

1. **(25pts)** The class **Animal** is started below. The **Movable** interface is completed below.

Complete the **Animal** class by doing the following:

* 1. Write an **overloaded constructor** so that it takes in 2 arguments. (1 for hasFur and 1 for numLegs)
  2. Write all the **getter/accessor** and **setter/mutator** methods.
  3. Write the **toString** method so that it returns a nicely formatted string including all the object’s values.

Use logic so that it will say “has fur” or “does not have fur” instead of “true” “false”

1. Write the **equals** method so that it takes in an Object, compares the calling object’s values agains the passed in and returns true if they all match or false if they do not.
2. Override the method **move** from the Movable interface so that it prints the way your animal moves.

|  |
| --- |
| public interface Movable {  public void move();  } |

|  |
| --- |
| public class Animal implements Movable{  private boolean hasFur;  private int numLegs;  public Animal(){  hasFur = true;  numLegs = 4;  }  } |

1. **(20pts)** The class **Sheep** is started for you below. Do the following to complete the class
   1. Add the necessary line to make the Sheep inherit from Animal
   2. Write an **overloaded constructor** so that it takes in 3 arguments. (favoriteFood, hasFur, numLegs)

Be sure to pass the appropriate arguments to the parent class’s constructor.

* 1. Override the **toString** method to include the details from the **Animal** class
  2. Override the **move** method so that it does NOT include the parent behavior, but instead prints “Jump over the Moon”

|  |
| --- |
| public class Sheep {  private String favoriteFood;  /\* The Default Constructor, and the getter, setter methods are created, but hidden \*/  } |

1. **(15 pts)** Write a **static** method named **printCrazyNums** that takes in an int and does the following:
   1. Starting from 50 less than the number and ending at that number Print on one line.
   2. All the even numbers as they are.
   3. All the odd numbers multiplied by 2.

Example: printCrazyNums(78);

Output: 28 58 30 62 32 66 34 70 36 74 38 …... 130 66 134 68 138 70 142 72 146 74 150 76 154 78

1. **(10 pts)** Write a **static** method named **timeForSomething** that interacts with the user via the console using the Scanner class. The method does NOT return a value.

The method should do the following:

Ask the user “Are you sleepy?”

As long as the user does not say “yes” with lower or upper case letters, the program should print out “TIME FOR FUN” and ask the question again.

When the user says “yes” with lower or upper case letters the program should print “Time for BED” and exit.