Free Response Questions (FRQs)

Solving FRQ Questions Shortcut

METHOD - Checklist
1. Read and understand the problem.
2. Return, cross it out if void or a constructor.
3. To do list. Plan out code beforehand in English, pseudocode.
4. Need variables? Declare **and** initialize.
5. Code.
6. Reread the problem (to ensure that the question was answered completely)

CLASS - Checklist
1. Read and understand
2. Class heading. Include extends/implements 1pt
3. private instance variables 1pt
4. Constructor, assign parameters 1pt
5. Other methods (treat as their own FRQs)
6. Reread the problem (to ensure that the question was answered completely)

Below are several hints for solving FRQs:

- First determine the type of question being asked. Do you need to write a method or a class?
    1. Most (about 75%) of the questions are method questions.

- **Methods**
    1. Read and understand the problem. You may have to read it several times. As you do this become familiar with the available attributes and methods for each class.

    2. Write the word return at the bottom of the page. If the method is void or a constructor, cross out the word return. This will remind you to either return, or keep you from accidentally returning a value.

    3. Create a To Do List. This will help you plan out your algorithm (strategy) for solving the problem. Don't be too vague and avoid restating the problem.

   4. Declare **and initialize** local variables at the top of the method. If you are unsure of what value to give the variable, give it the default value that it would have as an instance variable (attribute).

   5. Code the To Do List. At this point you may realize that you left a few items off of the list.

   6. Re-read the problem. Make sure that you answered the question asked. The prompt will often tell you how to handle special cases so make sure that you have addressed them. Also, make sure that your method works correctly for ALL of the examples.
   7.

- **Classes**
   1. Read and understand the problem. You may have to read it several times. As you do this become familiar with the available attributes and methods for each class.

   2. Write the method header; one point of the questions usually involves correct headers. Check to see if you need to use the keyword extends or implements.

   3. Create **private** instance variables. You will most likely lose a point if you do not declare your attributes as private.

   4. Write the constructor. The constructor's purpose is to assign values to the attributes. So look at the example and see how the constructor is being called. Look at the parameters; this will help you determine which attributes you need to initialize in the constructor.

   5. Remember : The attribute is on the left of the equal sign, the parameter is on the right.

   6. The first line of the constructor of an extended class must utilize the keyword **super**.

   7. Write the required methods. The prompt will tell you which methods you **must** have. Remember to override all abstract methods from interfaces.

   8. Use the **Method** strategy (found above) for each of the methods in the class.

General Tips

- The prompts usually give a To Do List in the comments above the method, the paragraph above the method, or as a bulleted list in the prompt. Look for these and develop a To Do List from them.

- If the method has a return type, you can often earn a point for:
  - Declaring a local variable of the same type
  - Initializing that local variable
  - Modifying that variable with the parameter(s) and/or attribute(s)
  - Returning that variable

- If the problem contains an array, List, or String, you will probably have to traverse it (loop through it).

- If you have a loop going through data then you can often earn a point for accessing each piece of that data. Ex:
  - Strings will probably utilize substring
  -
  - for(int a=0;a < s.length(); a++)
  - s.substring(a, a+1);
  -
  -
  - Arrays:
  -
  - for(int a=0;a < arr.length; a++)
  - arr[a];
  -
  -
  - Lists:
  -
  - for(int a=0;a < list.size(); a++)
  - list.get(a);

- While traversing an array, List, or String, make sure that you avoid out of bounds errors by "balancing out" the variables in the loop.

- Below is a standard way to print the values in an array:

- for(int a=0;a < arr.length; a++)
- System.out.println(arr[a]);

- The following will cause an out of bounds error when displaying the sum of two consecutive elements in the array.

- for(int a=0;a < arr.length; a++)
- System.out.println(arr[a]+arr[a+1]);
- To fix this, subtract one from the ending point of the for loop.

- for(int a=0;a < arr.length **-1**; a++)
- System.out.println(arr[a]+arr[a+1]);

- This is another way to cause an error while displaying the sum of two consecutive elements in the array.

- for(int a=0;a < arr.length; a++)
- System.out.println(arr[a-1]+arr[a]);
-
- To fix this, add one to the starting point of the for loop.
-
- for(int a=0**+1**;a < arr.length ; a++)
- System.out.println(arr[a-1]+arr[a]);
-
- However much you add to your loop variable when accessing data, subtract that amount from the ending point.
-
- However much you subtract from your loop variable when accessing data, add that amount to the starting point.
-
- for(int a=0**+x**;a < arr.length**-y** ; a++)
- System.out.println(arr[a**-x**]+arr[a**+y**]);
-
-
-
-
- If you have an array of Objects, you need to check for null values:
-
- for(int a=0;a < arr.length ; a++)

    - **if(arr[a]!=null)**
- System.out.println(arr[a].someMethod());
-
-
- Do not write an if/else that returns inside of a loop. It will only return the result of the first case.
-
- for(int a=0;a < arr.length ; a++)
- if(arr[a]%2==0)
- return true;
- else
- return false;
-
-
- When removing items from a List, there is a possibility to skip elements. If every element in this list was even, the loop would only remove half of the elements: for(int a=0;a < list.size() ; a++)
- if(list.get(a)%2==0)
- list.remove(a);

- There are two good ways to fix this problem:
    - This way decrements the iterator variable, so that it compensates for the shift caused by removing the element.
    - for(int a=0;a < list.size() ; a++)
    - if(list.get(a)%2==0)
    - list.remove(a);
    - a--;

    - This way traverses the list from the end to the beginning, which keeps the indexes of each elements from shifting.
    - for(int a=list.size()-1;a>=0;a--)
    - if(list.get(a)%2==0)
    - list.remove(a);

- When traversing an array, List, or String the iterator variable does not represent the element or a substring. It represents the index/ place in the String.

- Primitive variables cannot run methods. In the example:

- for(int a=0;a < list.size() ; a++)

- You cannot use a to run a substring method or utilize the compareTo method, etc.

- You could write:

- list.get(**a**).method();
-
-
- Do not use enhanced for loops (for each loops). You should not use enhanced for loops:
-
    - To remove elements as you traverse collections
    - To modify the current slot in an array or list
    - To iterate over multiple collections or arrays
-
-
- The AP Test generally does not ask you to write static methods in the class questions, because Java is an object-oriented language and static methods do not represent that well.
-
- Inappropriately using the keyword static has appeared as a point deduction on the rubric (see 2015 question 4).
-
-
- **Never write a recursive answer.** Most people don't do this correctly anyway.
-
-
- Most questions do not ask for output; check before you use a print statement.
-
-
- The AP Exam never asks for user input. Don't code any user input, ever.
-
-
- **Never hard-code an answer**; Many students try to code specifically to the examples. This type of answer typically earns few or no points.
-

- You should, however, apply your answer to each of the examples. Your answer should work correctly for each of the examples. If it does, then you most likely have a high scoring answer.
-
-
- Attributes (instance variables) are **always** private. Technically local variables should not have an access modifier, but College Board does not take any points off for doing so. If you get confused, make all variables private.
-
-
- Never alter the parameter or attributes, unless it is expressly stated in the prompt to do so.
-
- You may deep copy the data and alter that, but don't shallow copy it.

```
// Shallow Copy Example
ArrayList<String> originalList = new ArrayList<String>();
originalList.add("AP CS");
ArrayList<String> shallowCopyList = originalList; // This is a shallow copy.

// Deep Copy Example
ArrayList<String> deepCopyList = new ArrayList<String>(originalList.size());
for(String item : originalList) {
   deepCopyList.add(new String(item)); // This is a deep copy.
}
```

-
- Remember that Strings are immutable, so you may alter a String parameter, or a primitive, without losing the point for **destruction of persistent data**.

- Write neatly and stick to the Java subset for AP Computer Science. Utilizing esoteric methods and sloppy handwriting can cost you points if your grader can't follow your code.