AP Create Performance Task Template 2020-21

1. Program Code

Code Submission Checklist	*Yes or No?*
Did you capture all of your code and include comments acknowledging any code that was not written by you?	Yes
Did you add a comment on your button_click procedures (or other procedures that have input) marking them as input?	Yes
Did you show a list?	Yes
Did you show a procedure with a parameter and at least 2 calls to it?	Yes
Did you show selection and iteration in a procedure?	Yes

2. Video (created independently)

Video Submission Checklist	*Yes or No?*
Is your video under a minute and under 30 megabytes?	Yes
Did you make sure you did not talk, show your face, or identify yourself in your video?	Yes
Does the video show at least one working functionality element?	Yes
Does your program show an input (e.g. a button click) and an output (e.g. text, image, sound, movement, etc.?	Yes

3. Written Responses (created independently)

3a. Program Purpose and Function

i. Describe the overall purpose of the program

The overall purpose of the library management program is to inform students and librarians whether a certain book is available in the library or not. By entering the book title, it displays the availability of the book that the user searches for as an output.

ii. Describe what functionality of the program is demonstrated in the video

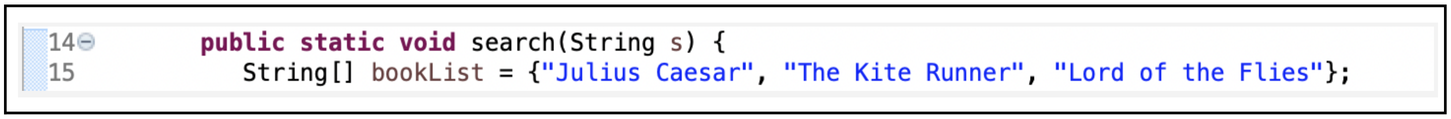
The functionality of the library management program demonstrated in the video is displaying a book is available if the user inputs a book title that is stored in the *bookList* and displaying that a book is not available if the user inputs a book title that is not stored in the *bookList*.

iii. Describe the input and output of the program demonstrated in the video

The input of the program is a string that indicates a book title that the user wants to search for its availability in the library. The output of the program is a text that indicates the result of the availability of the input book by checking its existence in the *bookList*.

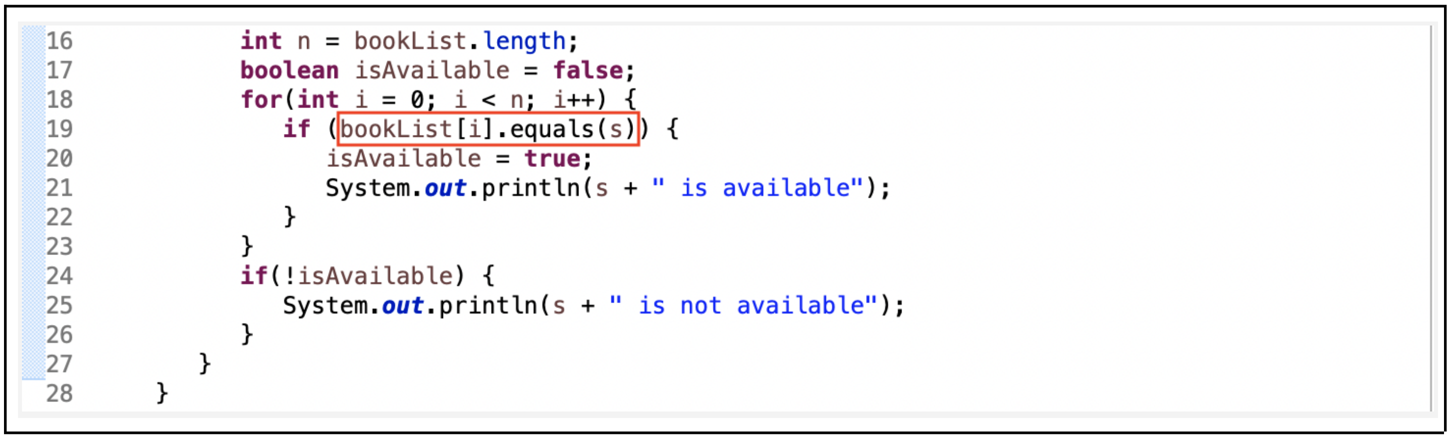
3b. List

i. The first program code segment must show how data have been stored in the list.



```java
        public static void search(String s) {
            String[] bookList = {"Julius Caesar", "The Kite Runner", "Lord of the Flies"};
```

ii. The second program code segment must show the data in the same list being used, such as creating new data from the existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.



```java
            int n = bookList.length;
            boolean isAvailable = false;
            for(int i = 0; i < n; i++) {
                if (bookList[i].equals(s)) {
                    isAvailable = true;
                    System.out.println(s + " is available");
                }
            }
            if(!isAvailable) {
                System.out.println(s + " is not available");
            }
        }
    }
```

iii. Identifies the name of the list being used in this response

The name of the list being used in this response is *bookList*.

iv. Describes what the data contained in the list represent in your program

The data contained in the *bookList* of this library management program represents the title of the books available in the library using this program. The *bookList* stores the title of the books the library contains as string data.

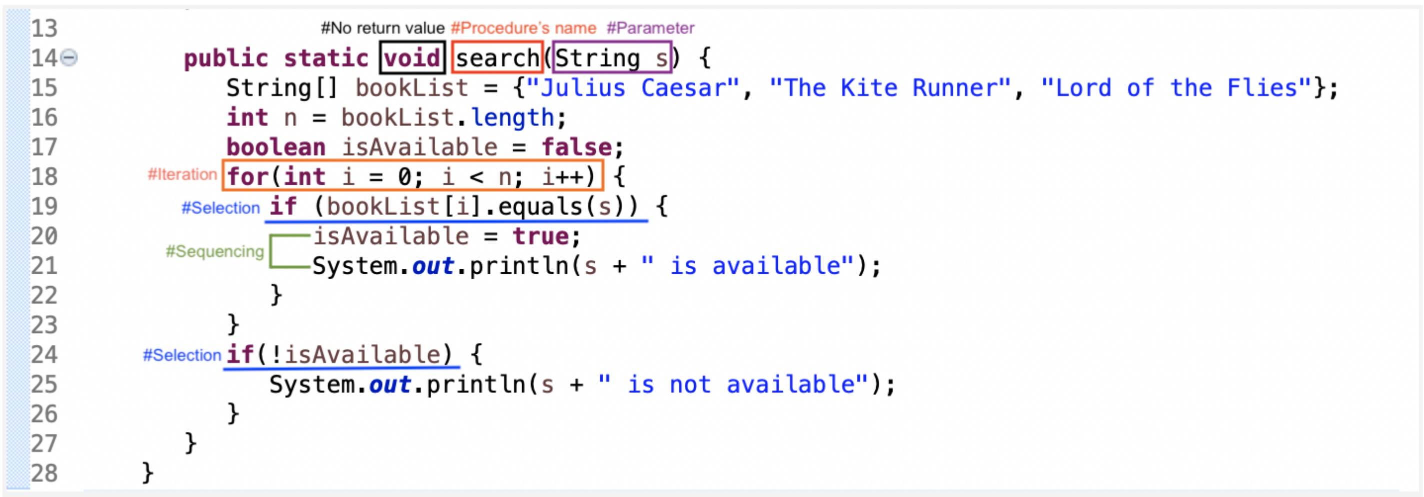
v. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list.
AP sidebar note: The data abstraction must make the program easier to develop (alternatives would be more complex) or easier to maintain (future changes to the size of the list would otherwise require significant modifications to the program code).

The *bookList* manages complexity in the library management program by allowing the book titles to be easily stored as strings in one set of data. Without the *bookList*, the program would be written to add every single book title as different string variables. Variables that store the book title in strings would have to be added to the program every single time when trying to add newly available books in the library without using a list. The *bookList* allows multiple strings that indicate book titles to be abstracted into one set of list data and contributes to managing the program's complexity.

3c. Procedure/Algorithms

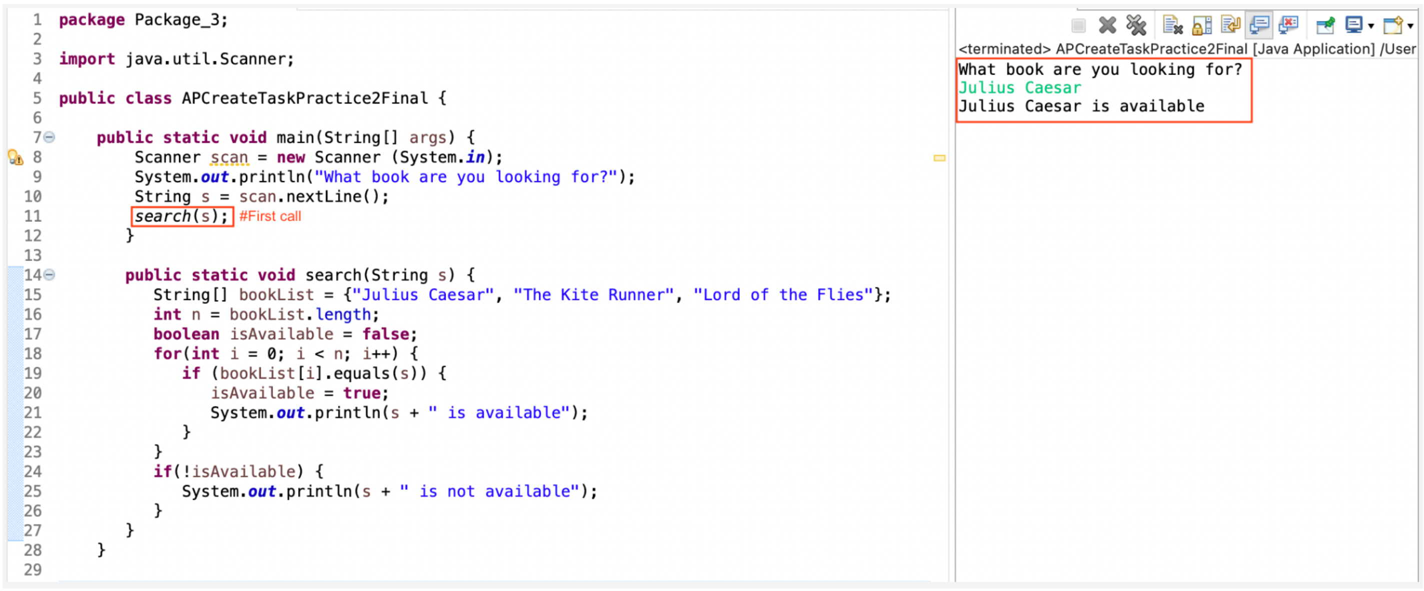
i. The first program code segment must be a student-developed procedure that:

- **Defines the procedure's name and return type (if necessary)**
- **Contains and uses one or more parameters that have an effect on the functionality of the procedure**
- **Implements an algorithm that includes sequencing, selection, and iteration**



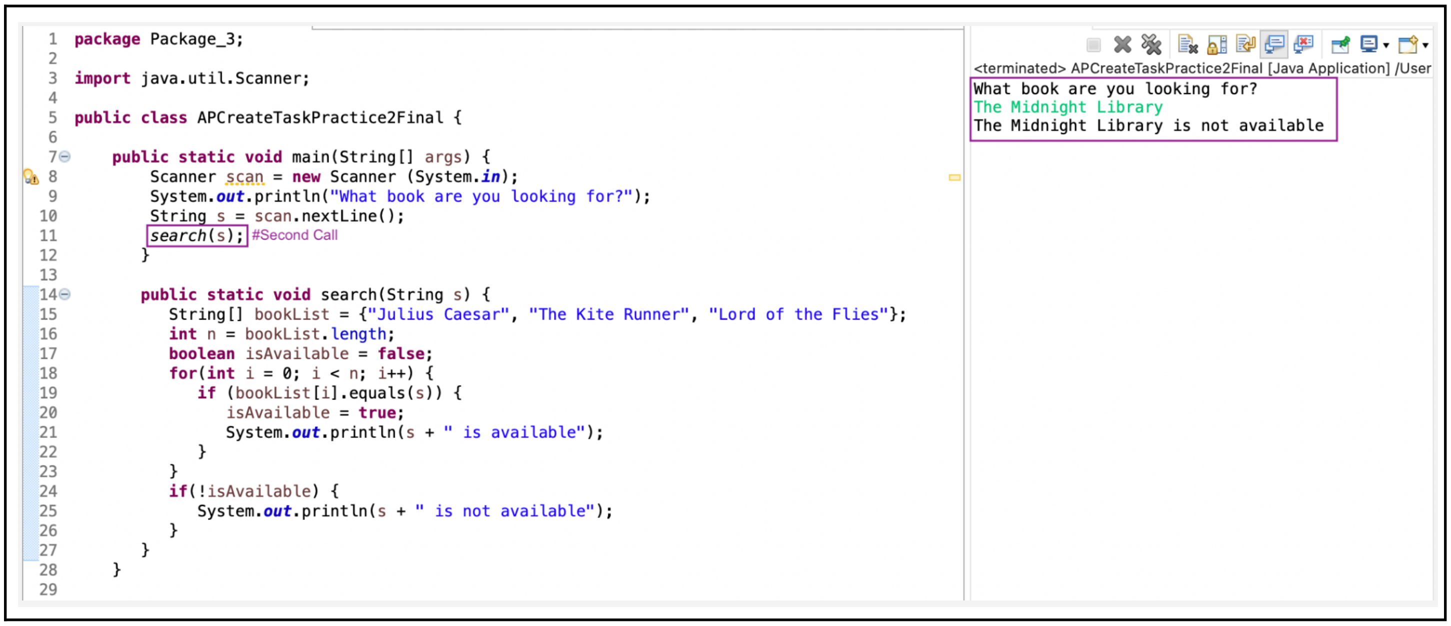
```
    public static void search(String s) {
        String[] bookList = {"Julius Caesar", "The Kite Runner", "Lord of the Flies"};
        int n = bookList.length;
        boolean isAvailable = false;
        for(int i = 0; i < n; i++) {
            if (bookList[i].equals(s)) {
                isAvailable = true;
                System.out.println(s + " is available");
            }
        }
        if(!isAvailable) {
            System.out.println(s + " is not available");
        }
    }
}
```

ii. The second program code segment must show where your student-developed procedure is being called in your program.



```
package Package_3;

import java.util.Scanner;

public class APCreateTaskPractice2Final {

    public static void main(String[] args) {
        Scanner scan = new Scanner (System.in);
        System.out.println("What book are you looking for?");
        String s = scan.nextLine();
        search(s);
    }

    public static void search(String s) {
        String[] bookList = {"Julius Caesar", "The Kite Runner", "Lord of the Flies"};
        int n = bookList.length;
        boolean isAvailable = false;
        for(int i = 0; i < n; i++) {
            if (bookList[i].equals(s)) {
                isAvailable = true;
                System.out.println(s + " is available");
            }
        }
        if(!isAvailable) {
            System.out.println(s + " is not available");
        }
    }
}
```

```
<terminated> APCreateTaskPractice2Final [Java Application] /User
What book are you looking for?
Julius Caesar
Julius Caesar is available
```



```java
package Package_3;

import java.util.Scanner;

public class APCreateTaskPractice2Final {

    public static void main(String[] args) {
        Scanner scan = new Scanner (System.in);
        System.out.println("What book are you looking for?");
        String s = scan.nextLine();
        search(s); #Second Call
       }

       public static void search(String s) {
          String[] bookList = {"Julius Caesar", "The Kite Runner", "Lord of the Flies"};
          int n = bookList.length;
          boolean isAvailable = false;
          for(int i = 0; i < n; i++) {
             if (bookList[i].equals(s)) {
                isAvailable = true;
                System.out.println(s + " is available");
             }
          }
          if(!isAvailable) {
             System.out.println(s + " is not available");
          }
       }
    }
```

```
<terminated> APCreateTaskPractice2Final [Java Application] /User
What book are you looking for?
The Midnight Library
The Midnight Library is not available
```

iii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program.

The identified procedure *search* takes the user input as a string parameter and stores it as a variable using a scanner. Then, it compares that variable with each of the data stored in the *bookList* and displays whether or not the book the user is searching for is available in the library. The procedure named *search* contributes to the program's overall functionality by comparing the input data with every element in the list and displaying the result of the availability as an output, which is the critical function of this library management program.

iv. Explains in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

The algorithm in the identified procedure *search* initially displays the question “What book are you looking for?” to the user’s screen as a text. Then, when the user types the book title that he or she wants to search for its availability in the library as an answer, the procedure receives that string input as a parameter and stores it as a string variable using a scanner. After that, it compares the string variable with each string data stored in the list. If the exact string exists in the *bookList*, it displays that the book the user is looking for is available. If the exact string does not exist in the *bookList*, it displays that the book the user is looking for is not available.

3d. Procedure Calls

i. Describes two calls to the procedure identified in written response 3c.
Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.
First call *(describe in code and/or written text the procedure call and its argument)***:**

In the first call, I tested the book title that existed in the *bookList* of the program code. I ran the program and typed the book title "Julius Caesar", which was already stored in the *bookList*. Since the element in the *bookList* was equal to the string input typed, the *isAvailable* boolean that was initially false changed to true, leading the program to display "Julius Caesar is available".

Second call *(describe in code and/or written text, the procedure call and its argument, which must be different than the argument in the first call)***:**

In the second call, I tested the book title that did not exist in the *bookList* of the program code. I ran the program and typed the book title "The Midnight Library", which was not stored as an element of the *bookList*. Since no element in the *bookList* was equal to the string input typed, the *isAvailable* boolean remained false, leading the program to display "The Midnight Library is not available".

ii. Describes what condition(s) is being tested by each call to the procedure
Condition(s) tested by the first call *(describe how this procedure call is being used in your program and its purpose)*

The condition tested by the first call is the existence of an element in the *bookList* that is equal to the string input.

Condition(s) tested by the second call *(describe how the second procedure call is used in your program and its purpose)*

The condition tested by the second call is the existence of an element in the *bookList* that is equal to the string input.

iii. Identifies the result of each call
Result of the first call: *(describe what happens and what output is produced when this procedure call runs in your program)*

The result of the first call changed the boolean *isAvailable* to true and displayed "Julius Caesar is available".

Result of the second call: *(describe what happens and what output is produced when the second procedure call runs in your program)*

The result of the second call made the boolean *isAvailable* to remain false and displayed "The Midnight Library is not available".