Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position `index (0 <= index <= size)`, moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

Free-Response Section

Scoring Guidelines

Applying the Scoring Criteria

Apply the question scoring criteria first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

v) Array/collection access confusion (`[] get`)

w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

x) Local variables used but none declared

y) Destruction of persistent data (e.g., changing value referenced by parameter)

z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ ≤ ≥ <> ≠)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `( )`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `( )` on parameter-less method or constructor invocations
- Missing `( )` around `if` or `while` conditions

Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be **unambiguously inferred from context, for example, "ArayList" instead of "ArrayList". As a counterexample, note that if the code declares `"int G=99, g=0;"`, then uses `"while (G < 10)"` instead of `"while (g < 10)"`, the context does **not** allow for the reader to assume the use of the lower case variable.*

AP® COMPUTER SCIENCE A
2008 SCORING GUIDELINES

Question 1: Flight List

Part A: getDuration **4 points**

+1 handle empty case
- **+1/2** check if `flights` is empty
- **+1/2** return 0 if empty

+1 access start time
- **+1/2** access `flights.get(0)`
- **+1/2** correctly call `getDepartureTime` on a flight

+1 access end time
- **+1/2** access `flights.get(flights.size()-1)`
- **+1/2** correctly call `getArrivalTime` on a flight

+1 calculate and return duration
- **+1/2** call `minutesUntil` using `Time` objects
- **+1/2** return correct duration (using `minutesUntil`)

Part B: getShortestLayover **5 points**

+1 handle case with 0 or 1 flight
- **+1/2** check if `flights.size() < 2`
- **+1/2** return -1 in that case

+1 traverse `flights`
- **+1/2** correctly access an element of `flights` (in context of loop)
- **+1/2** access all elements of `flights` (lose this if index out-of-bounds)

+2 1/2 find shortest layover (in context of loop)
- **+1** get layover time between successive flights (using `minutesUntil`)
- **+1/2** compare layover time with some previous layover
- **+1** correctly identify shortest layover

+1/2 return shortest layover

AP® COMPUTER SCIENCE A
2008 CANONICAL SOLUTIONS

Question 1: Flight List

PART A:

```
public int getDuration()
{
  if (flights.size() == 0)
  {
    return 0;
  }
  else
  {
    Time start = flights.get(0).getDepartureTime();
    Time end = flights.get(flights.size()-1).getArrivalTime();
    return start.minutesUntil(end);
  }
}
```

PART B:

```
public int getShortestLayover()
{
  if (flights.size() < 2)
  {
    return -1;
  }
  else
  {
    int shortest = getDuration();
    for (int i = 0; i < flights.size()-1; i++)
    {
      Time arrive = flights.get(i).getArrivalTime();
      Time leave = flights.get(i+1).getDepartureTime();
      int layover = arrive.minutesUntil(leave);
      if (layover < shortest)
      {
        shortest = layover;
      }
    }
    return shortest;
  }
}
```