

APPENDIX B

Standard Java Library Methods Required for AP CS A

Accessible Methods from the Java Library That May Be Included on the Exam

```
class java.lang.Object
• boolean equals(Object other)
• String toString()

interface java.lang.Comparable*
• int compareTo(Object other) // returns a value < 0 if this is less than other
// returns a value = 0 if this is equal to other
// returns a value > 0 if this is greater than other

class java.lang.Integer implements java.lang.Comparable*
• Integer(int value)
• int intValue()

class java.lang.Double implements java.lang.Comparable*
• Double(double value)
• double doubleValue()

class java.lang.String implements java.lang.Comparable*
• int length()
• String substring(int from, int to) // returns the substring beginning at from
// and ending at 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

class java.lang.Math
• static int abs(int x)
• static double abs(double x)
• static double pow(double base, double exponent)
• static double sqrt(double x)
• static double random() // returns a double in the range [0.0, 1.0)

class java.util.ArrayList<E>
• int size()
• 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 adjusts size
• E get(int index)
• E set(int index, E obj) // replaces the element at position index with obj
// returns the element formerly at the specified position
• 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 adjusts size
// returns the element formerly at the specified position
```

* The AP Java subset uses the "raw" Comparable interface, not the generic Comparable<T> interface.

APPENDIX C

Standard Java Library Methods Required for AP CS AB

Note: The `java.util.Stack` and `java.util.PriorityQueue` classes and the `java.util.Queue` interface (page 109 in this Appendix) each inherit methods that access elements in a way that violates their abstract data structure definitions. Solutions that use objects of types `Stack`, `Queue`, and `PriorityQueue` should use only the methods listed in the Appendix for accessing and modifying those objects. The use of other methods in free-response solutions may not receive full credit.

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

class java.lang.Object

- `boolean equals(Object other)`
- `String toString()`
- `int hashCode()`

interface java.lang.Comparable*

- `int compareTo(Object other)` // returns a value < 0 if `this` is less than `other`
// returns a value $= 0$ if `this` is equal to `other`
// returns a value > 0 if `this` is greater than `other`

class java.lang.Integer implements java.lang.Comparable*

- `Integer(int value)`
- `int intValue()`

class java.lang.Double implements java.lang.Comparable*

- `Double(double value)`
- `double doubleValue()`

class java.lang.String implements java.lang.Comparable*

- `int length()`
- `String substring(int from, int to)` // returns the substring beginning at `from`
// and ending at `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

class java.lang.Math

- `static int abs(int x)`
- `static double abs(double x)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double x)`
- `static double random()` // returns a `double` in the range `[0.0, 1.0)`

* The AP Java subset uses the "raw" `Comparable` interface, not the generic `Comparable<T>` interface.

```

interface java.util.List<E>
• int size()
• 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 adjusts size

• E get(int index)
• E set(int index, E obj) // replaces the element at position index with obj
// returns the element formerly at the specified position
• 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 adjusts size
// returns the element formerly at the specified position

• Iterator<E> iterator()
• ListIterator<E> listIterator()

class java.util.ArrayList<E> implements java.util.List<E>

class java.util.LinkedList<E> implements java.util.List<E>, java.util.Queue<E>
• void addFirst(E obj)
• void addLast(E obj)
• E getFirst()
• E getLast()
• E removeFirst()
• E removeLast()

interface java.util.Queue<E> // implemented by LinkedList<E>
• boolean add(E obj) // enqueues obj at the end of the queue; returns true
• E remove() // dequeues and returns the element at the front of the queue
• E peek() // returns the element at the front of the queue;
// null if the queue is empty

• boolean isEmpty()

class java.util.PriorityQueue<E> // E should implement Comparable*
• boolean add(E obj) // adds obj to the priority queue; returns true
• E remove() // removes and returns the minimal element from the priority queue
• E peek() // returns the minimal element from the priority queue;
// null if the priority queue is empty

• boolean isEmpty()

class java.util.Stack<E>
• E push(E item) // pushes item onto the top of the stack; returns item
• E pop() // removes and returns the element at the top of the stack
• E peek() // returns the element at the top of the stack;
// throws an exception if the stack is empty

• boolean isEmpty()

```

* The AP Java subset uses the "raw" Comparable interface, not the generic Comparable<T> interface.

```

interface java.util.Iterator<E>
• boolean hasNext()
• E next()
• void remove() // removes the last element that was returned by next

interface java.util.ListIterator<E> extends java.util.Iterator<E>
• void add(E obj) // adds obj before the element that will be returned by next
• void set(E obj) // replaces the last element returned by next with obj

interface java.util.Set<E>
• int size()
• boolean contains(Object obj)
• boolean add(E obj) // if obj is not present in this set, adds obj and returns true;
// otherwise, returns false
• boolean remove(Object obj) // if obj is present in this set, removes obj and returns true;
// otherwise, returns false
• Iterator<E> iterator()

class java.util.HashSet<E> implements java.util.Set<E>
class java.util.TreeSet<E> implements java.util.Set<E>

interface java.util.Map<K,V>
• int size()
• boolean containsKey(Object key)
• V put(K key, V value) // associates key with value
// returns the value formerly associated with key
// or null if key was not present
• V get(Object key) // returns the value associated with key
// or null if there is no associated value
• V remove(Object key) // removes and returns the value associated with key;
// returns null if there is no associated value
• Set<K> keySet()

class java.util.HashMap<K,V> implements java.util.Map<K,V>
class java.util.TreeMap<K,V> implements java.util.Map<K,V>

```