AP® Computer Science AB
2004 Sample Student Responses

AB 1 A1

1. Consider designing and implementing a set of classes and interfaces to represent books, library items, and library books.

   (a) A library item contains two pieces of information: a unique ID (a `String`) and a holder (also a `String`). If the library item has been checked out, the holder is the name of the person who has checked out this item, otherwise the holder is `null`. The ID of a library item cannot change after it is created, but the holder can change. Write the `LibraryItem` interface that abstracts this functionality.

```
public interface LibraryItem
{
    String getIDC);
    String getHolder();
    void changeHolder( String newHolder );
}
```

Part (b) begins on page 6.

A2

(b) A book consists of an author and a title, neither of which can change once the book is created. The `Book` class is represented as follows.

```
public class Book
{
   [redacted]

   [redacted]
   { theAuthor = author;  theTitle = title; }

   [redacted]
   { return theAuthor; }

   [redacted]
   { return theTitle; }

}
```

A library book is a book that is also a library item. Write the complete class `LibraryBook`, implementing the required methods.

```
public class LibraryBook extends Book implements LibraryItem
{
    public LibraryBook(String author, String title, String id)
    {
        super(author, title);
        theID = id;
        theHolder = null;
    }
    public String getID()
    { return theID; }
    public String getHolder()
    { return theHolder; }
    public void changeHolder(String newHolder)
    { theHolder = newHolder; }
    private String theID;
    private String theHolder;
}
```

GO ON TO THE NEXT PAGE.

A3

(c) A library is a collection of library items and supports the following operations.

1. Add a library item to the library.
2. Check out a library item by specifying its ID and new holder.
3. Determine the current holder of a library item, given its ID.

Consider the following incomplete class declaration.

```
public class Library
{
   private SomeClass libraryItems;

   public Library()
   {  /* implementation not shown */  }

   public void add(LibraryItem item)
   {  /* implementation not shown */  }

   public void checkout(String id, String newHolder)
   {  /* implementation not shown */  }

   public String getHolder(String id)
   {  /* implementation not shown */  }
}
```

Choose a data representation for ***SomeClass*** from the `java.util` class library that allows the operations `add`, `checkout`, and `getHolder` to be performed efficiently. In doing so, you must explain how you are using your chosen `java.util` class to organize your data and give the expected Big-Oh running time for each of these three operations in terms of `n`, the number of items in the library.

Your data representation (Circle one)		How data is organized
ArrayList	LinkedList	Use IDs as keys mapping to LibraryItems.
(HashMap)	TreeMap	
HashSet	TreeSet	

Operations	Expected Big-Oh efficiency
add	O(1)
checkout	O(1)
getHolder	O(1)

GO ON TO THE NEXT PAGE.

AB1 B1

1. Consider designing and implementing a set of classes and interfaces to represent books, library items, and library books.

   (a) A library item contains two pieces of information: a unique ID (a `String`) and a holder (also a `String`). If the library item has been checked out, the holder is the name of the person who has checked out this item, otherwise the holder is `null`. The ID of a library item cannot change after it is created, but the holder can change. Write the `LibraryItem` interface that abstracts this functionality.

```
public interface LibraryItem
{
    String getID();
    String getHolder();
    void setHolder(String h);
}
```

Part (b) begins on page 6.

(b) A book consists of an author and a title, neither of which can change once the book is created. The `Book` class is represented as follows.

```
public class Book
{
   [redacted]

   [redacted]
   { theAuthor = author;  theTitle = title; }

   [redacted]
   { return theAuthor; }

   [redacted]
   { return theTitle; }

}
```

A library book is a book that is also a library item. Write the complete class `LibraryBook`, implementing the required methods.

```
public class LibraryBook extends LibraryItem
{
  public String myID;
  public String myHolder;
  Public Book bk;

    public LibraryBook (Book novel, String id)
    {
       myID = id;
       myHolder = null;
       bk = novel;
    }
    public String getID()
    {
      return (myID);
    }
    public String getHolder()
    {
      return (myHolder);
    }
    public void setHolder (String h)
    {
      myHolder = h;
    }
}
```

(c) A library is a collection of library items and supports the following operations.

1. Add a library item to the library.
2. Check out a library item by specifying its ID and new holder.
3. Determine the current holder of a library item, given its ID.

Consider the following incomplete class declaration.

```
public class Library
{
  [redacted]

  [redacted]
  {  /* implementation not shown */  }

  [redacted]
  {  /* implementation not shown */  }

  [redacted]
  {  /* implementation not shown */  }

  [redacted]
  {  /* implementation not shown */  }
}
```

Choose a data representation for ***SomeClass*** from the `java.util` class library that allows the operations `add`, `checkout`, and `getHolder` to be performed efficiently. In doing so, you must explain how you are using your chosen `java.util` class to organize your data and give the expected Big-Oh running time for each of these three operations in terms of `n`, the number of items in the library.

Your data representation (Circle one)		How data is organized
`ArrayList`	`LinkedList`	LibraryItems are stored in a HashSet using id to create a Hash key value.
`HashMap`	`TreeMap`	
(`HashSet`)	`TreeSet`	

Operations	Expected Big-Oh efficiency
`add`	O(1)
`checkout`	O(1)
`getHolder`	O(1)

GO ON TO THE NEXT PAGE

AB1 C.

1. Consider designing and implementing a set of classes and interfaces to represent books, library items, and library books.

   (a) A library item contains two pieces of information: a unique ID (a `String`) and a holder (also a `String`). If the library item has been checked out, the holder is the name of the person who has checked out this item, otherwise the holder is `null`. The ID of a library item cannot change after it is created, but the holder can change. Write the `LibraryItem` interface that abstracts this functionality.

```
LibraryItem Interface
{
    public LibraryItem(String id, String hold)
    public String ID, holder;
    public void changeholder(String nuholder)
    public String getHolder()
    public String getID()
}
```

Part (b) begins on page 6.

GO ON TO THE NEXT PAGE.

(b) A book consists of an author and a title, neither of which can change once the book is created. The `Book` class is represented as follows.

```
public class Book
{
   [redacted]
   [redacted]

   [redacted]
   { theAuthor = author;  theTitle = title; }

   [redacted]
   { return theAuthor; }

   [redacted]
   { return theTitle; }

}
```

A library book is a book that is also a library item. Write the complete class `LibraryBook`, implementing the required methods.

```
class LibraryBook extends Book, implements LibraryItem
{
    public LibraryBook(String author, String title)
    {
        super(author, title);
    }
}
```

C3

(c) A library is a collection of library items and supports the following operations.

1. Add a library item to the library.
2. Check out a library item by specifying its ID and new holder.
3. Determine the current holder of a library item, given its ID.

Consider the following incomplete class declaration.

```
public class Library
{
   private SomeClass items;

   public Library()
   {  /* implementation not shown */  }

   public void addItem(LibraryItem newItem)
   {  /* implementation not shown */  }

   public void checkout(int itemID, String newHolder)
   {  /* implementation not shown */  }

   public String getHolder(int itemID)
   {  /* implementation not shown */  }
}
```

Choose a data representation for ***SomeClass*** from the `java.util` class library that allows the operations `add`, `checkout`, and `getHolder` to be performed efficiently. In doing so, you must explain how you are using your chosen `java.util` class to organize your data and give the expected Big-Oh running time for each of these three operations in terms of `n`, the number of items in the library.

Your data representation (Circle one)		How data is organized
ArrayList	LinkedList	data organized by last 3 #'s of ID#
HashMap	TreeMap	
~~HashSet~~	TreeSet	

Operations	Expected Big-Oh efficiency
add	O(1)
checkout	O(1)
getHolder	O(1)

GO ON TO THE NEXT PAGE.