CS 61B Midterm 1 Review

Dan Wang, Jonathan Lin, Sung Roa Yoon, Edwin Liao

Eta Kappa Nu, Mu Chapter
University of California, Berkeley

19 February 2012

True/False

The two loops print the same number of lines:

```java
int i = 0;
while (i < 5) {
    System.out.println();
    i++;
}
```

```java
int j = 0;
for(; j < 5; j++) {
    System.out.println();
}
```

True/False

The two loops print the same number of lines:

```java
int i = 0;
while (i < 5) {
    System.out.println();
    i++;
}
```

```java
int j = 0;
for(; j < 5; j++) {
    System.out.println();
}
```

True

True/False

This is a valid Java statement:

```java
int[][][] array = new int[][][];
```

True/False

This is a valid Java statement:

```java
int[][][] array = new int[][][];
```

False

True/False

This is a valid Java statement:

```java
int[][][] array = new int[][][];
```

False

This is a valid Java statement:

```java
int[][][] array = new int[3][][];
```

True/False

This is a valid Java statement:

```java
int[][][] array = new int[][][];
```

False

This is a valid Java statement:

```java
int[][][] array = new int[3][][];
```

True

Multiple Choice

What is the value of `k` at the end?

```java
int[] array = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int k = 0;
for (; k < array.length; k++) {
  k += array[k];
}
```

1. 10
2. 11
3. 15
4. 55

Multiple Choice

What is the value of `k` at the end?

```java
int[] array = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int k = 0;
for (; k < array.length; k++) {
  k += array[k];
}
```

1. 10
2. 11
3. 15
4. 55

Multiple Choice

Select the assertions that are true after execution of the following method:

```
public static void main(String[] args) {
    String s1 =  Hello World ;
    String s2 = s1;
    String s3 = new String(s1);
}
```

1. s1 == s2
2. s2 == s3
3. s1.equals(s2)
4. s2.equals(s3)
5. Error

Multiple Choice

Select the assertions that are true after execution of the following method:

```
public static void main(String[] args) {
    String s1 = "Hello World";
    String s2 = s1;
    String s3 = new String(s1);
}
```

1. `s1 == s2`
2. `s2 == s3`
3. `s1.equals(s2)`
4. `s2.equals(s3)`
5. Error

Multiple Choice

What is printed after the following code is executed?

```
public static void main(String[] args) {
    Robot chell = new Robot();
    String s = "hello human";
    chell.cake(s);
    System.out.print(s + " " + chell.s);
}
public Robot() {
    String s = "I have a cake";
    public void cake(String s) {
        this.s = "cake is a lie";
        s = "bye human";
    }
}
```

Multiple Choice

1. `"hello human cake is a lie"`
2. `"hello human I have a cake"`
3. `"cake is a lie I have a cake"`
4. `"bye human I have a cake"`
5. `"hello human bye human"`

Multiple Choice

1. `"hello human cake is a lie"`
2. `"hello human I have a cake"`
3. `"cake is a lie I have a cake"`
4. `"bye human I have a cake"`
5. `"hello human bye human"`

Multiple Choice

What is printed after the following code is executed?

```
public static void main(String[] args) {
    America Bob = new America();
    America Mary = new America();
    Bob.earnMoney(100);
    Mary.earnMoney(1000);
    System.out.println(America.publicDebt);
}
public America() {
    int myMoney = 0;
    static int publicDebt = 1000000; //1,000,000
    public void earnMoney(int wage) {
        myMoney += wage;
        publicDebt += wage * 1000;
    }
}
```

Multiple Choice

1. 2000000
2. 1100000
3. 2100000
4. 1000000
5. Error

Multiple Choice

1. 2000000
2. 1100000
3. 2100000
4. 1000000
5. Error

Multiple Choice

What is printed after the following code is executed?

```
public static void main(String[] args) {
    myWallet Bob = new myWallet();
    Bob.earnMoney();
    if (Bob.hasMoney) {
        System.out.println("Yay!");
    } else {
        System.out.println("Awww");
    }
}
public myWallet() {
    boolean hasMoney = false;
    public static void earnMoney() {
        hasMoney = true;
    }
}
```

Multiple Choice

1. `"Yay!"`
2. `"Awww"`
3. Error

Multiple Choice

1. `"Yay!"`
2. `"Awww"`
3. Error

Multiple Choice

What is printed after the following code is executed?

```java
public static void main(String[] args) {
    int a = 5;
    int b = 4;
    int c = a / b;
    System.out.println(c);
}
```

Multiple Choice

1. `1.25`
2. `1`
3. `2`
4. Error

Multiple Choice

1. `1.25`
2. 1
3. 2
4. Error

Multiple Choice

What is printed after the following code is executed?

```java
public static void main(String[] args) {
    int a = 5;
    double b = 4;
    double c = a / b;
    System.out.println(c);
}
```

Multiple Choice

1. `1.25`
2. `1`
3. `2`
4. Error

Multiple Choice

1. 1.25
2. 1
3. 2
4. Error

Multiple Choice

What is printed after the following code is executed?

```java
public static void main(String[] args) {
  double a = 5.5;
  int b = a;
  System.out.println(b);
}
```

Multiple Choice

1. `5.5`
2. 5
3. 6
4. Error

Multiple Choice

1. `5.5`
2. `5`
3. `6`
4. Error

Fill in the blanks

Fill in the following function:

```
boolean isSorted(int[] array) {
    // array: array to test if sorted if not
    // Returns: true if array is sorted in increasing
    // order

    for (int i = ___; ________; i++)
        if (array[___] < array[___]) {
            return false;
        }
    return true;
}
```

Fill in the blanks

Solution:

```
boolean isSorted(int[] array) {
    for (int i = 1; i < array.length; i++)
        if (array[i] < array[i - 1]) {
            return false;
        }
    return true;
}
```

Fill in the blanks

Fill in the following function:

```
boolean isPalindrome(int[] array) {
    // Returns: true if the array is a palindrome

    int lower = ___, upper = ___;

    while (___) {
        if (array[lower] != array[upper]) {
            return false;
        }
        lower++;
        upper--;
    }

    return true;
}
```

Fill in the blanks

Solution:

```
boolean isPalindrome(int[] array) {
    int lower = 0, upper = array.length - 1;

    while (lower <= upper) {
        if (array[lower] != array[upper]) {
            return false;
        }
        lower++;
        upper--;
    }

    return true;
}
```

Fill in the blanks

Fill in the following function:

```
boolean hasPalindrome(int[] array, int length) {
  // Returns: true if array contains a palindrome
  // of length at least i

  for (int i = 0; i <= _____; i++) {
    for (int j = _____; j <= _____; j++) {
      if (isPalindrome(Arrays.copyOfRange(array, i, j))) {
        return true;
      }
    }
  }

  return false;
}
```

Fill in the blanks

Solution:

```
boolean hasPalindrome(int[] array, int length) {
  for (int i = 0; i <= array.length - length; i++) {
    for (int j = i + length; j <= array.length; j++) {
      if (isPalindrome(Arrays.copyOfRange(array, i, j))) {
        return true;
      }
    }
  }
  return false;
}
```

Fill in the blanks

Fill in the following function:

```
int largestPalindrome(int[] array) {
    // Returns: the length of the longest
    // palindrome in the array

    for (int i = ___; ___; ___) {
        if (_____) {
            return i;
        }
    }

    return 0;
}
```

Fill in the blanks

Solution:

```
int largestPalindrome(int[] array) {
    for (int i = array.length; i >= 1; i--) {
        if (hasPalindrome(array, i)) {
            return i;
        }
    }

    return 0;
}
```

What will be printed?

What is printed after the following code is executed?

```java
public static void main(String[] args) {
    String s = "Is this the real life?";
    change(s);
    System.out.println(s);
}
public static void change(String s) {
    s = "Is this just fantasy?";
}
```

1. Is this the real life?
2. Is this just fantasy?
3. s
4. Error

What will be printed?

What is printed after the following code is executed?

```
public static void main(String[] args) {
    String s = "Is this the real life?";
    change(s);
    System.out.println(s);
}
public static void change(String s) {
    s = "Is this just fantasy?";
}
```

1. Is this the real life?
2. Is this just fantasy?
3. s
4. Error

What will be printed?

What is printed after the following code is executed?

```
public static void main(String[] args) {
    int[] arr = {1, 2, 3};
    change(arr);
    System.out.println(arr[0]);
}
public static void change(int[] i) {
    i[0] = 5;
    i = null;
}
```

1. 1
2. 5
3. null
4. error

What will be printed?

What is printed after the following code is executed?

```
public static void main(String[] args) {
    int[] arr = {1, 2, 3};
    change(arr);
    System.out.println(arr[0]);
}
public static void change(int[] i) {
    i[0] = 5;
    i = null;
}
```

1. 1
2. 5
3. null
4. error

What will be printed?

What is printed after the following code is executed?

```java
public static void main(String[] args) {
    int herp = 4;
    int derp = 6;
    herp = derp;
    herp = herp + 1;
    System.out.println(derp);
}
```

1. 4
2. 6
3. 5
4. 7

What will be printed?

What is printed after the following code is executed?

```java
public static void main(String[] args) {
     int herp = 4;
     int derp = 6;
     herp = derp;
     herp = herp + 1;
     System.out.println(derp);
}
```

1. 4
2. 6
3. 5
4. 7

What will be printed?

What is printed after the following code is executed?

```java
public static void main(String[] args) {
    String x = "Caught in a landslide,";
    String y = "No escape from reality";
    String z = x;
    x = y;
    System.out.println(z);
}
```

1. Caught in a landslide,
2. No escape from reality
3. null
4. Error

What will be printed?

What is printed after the following code is executed?

```
public static void main(String[] args) {
    String x = "Caught in a landslide,";
    String y = "No escape from reality";
    String z = x;
    x = y;
    System.out.println(z);
}
```

1. Caught in a landslide,
2. No escape from reality
3. null
4. Error

What will be printed?

What is printed after the following code is executed?

```java
Panda p = new Panda();
Animal a = p;
boolean wat = (a == p);
System.out.println(wat);
```

1. true
2. false
3. wat
4. Error

What will be printed?

What is printed after the following code is executed?

```java
Panda p = new Panda();
Animal a = p;
boolean wat = (a == p);
System.out.println(wat);
```

1. true
2. false
3. wat
4. Error

Overriding Methods

Assume that Subclass is a subclass of Class and that Class has the following method defined:

```
class Class {
    public void foo(int x) { ... }
}
```

True or false: The following method in Subclass overrides Class's foo().

```
class Subclass extends Class {
    public int foo(int y) { ... }
}
```

1. True
2. False

Overriding Methods

Assume that `Subclass` is a subclass of `Class` and that `Class` has the following method defined:

```
class Class {
    public void foo(int x) { ... }
}
```

True or false: The following method in `Subclass` overrides `Class`'s `foo()`.

```
class Subclass extends Class {
    public int foo(int y) { ... }
}
```

1. True
2. False

Overriding Methods

Assume that `Subclass` is a subclass of `Class` and that `Class` has the following method defined:

```
class Class {
    public void foo(Object o) { ... }
}
```

True or false: The following method in `Subclass` overrides `Class`'s `foo()`.

```
class Subclass extends Class {
    public void foo(String s) { ... }
}
```

1. True
2. False

Overriding Methods

Assume that `Subclass` is a subclass of `Class` and that `Class` has the following method defined:

```java
class Class {
    public void foo(Object o) { ... }
}
```

True or false: The following method in `Subclass` overrides `Class`'s `foo()`.

```java
class Subclass extends Class {
    public void foo(String s) { ... }
}
```

1. True
2. False

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Class c = new Class();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Class c = new Class();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Subclass c = new Class();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Subclass c = new Class();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Class c = new Subclass();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that Subclass is a subclass of Class and do_something is a public non-static method in both classes.

```
Class c = new Subclass();
c.do_something();
```

1. Class's method is called
2. Subclass's method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Object c = new Class();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `do_something` is a `public` non-static method in both classes.

```
Object c = new Class();
c.do_something();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `some_value` is a `public` field in both classes.

```
Class c = new Subclass();
System.out.println(c.some_value);
```

1. `Class`'s field is printed
2. `Subclass`'s field is printed
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `some_value` is a `public` field in both classes.

```
Class c = new Subclass();
System.out.println(c.some_value);
```

1. `Class`'s field is printed
2. `Subclass`'s field is printed
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `static_value` is a **public static** field in both classes.

```
Class c = new Subclass();
System.out.println(c.static_value);
```

1. `Class`'s field is printed
2. `Subclass`'s field is printed
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `static_value` is a **public static** field in both classes.

```
Class c = new Subclass();
System.out.println(c.static_value);
```

1. `Class`'s field is printed
2. `Subclass`'s field is printed
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `static_method()` is a **public static** method in both classes.

```
Class c = new Subclass();
c.static_method();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

What will happen?

What will happen when the following code is run? Assume that `Subclass` is a subclass of `Class` and `static_method()` is a **public static** method in both classes.

```
Class c = new Subclass();
c.static_method();
```

1. `Class`'s method is called
2. `Subclass`'s method is called
3. Compile-time error
4. Run-time error

General Rule

In general, if we define a variable `var` as such:

```
// S and D are predefined classes
S var = new D();
S.X;
```

Then `S` is the **static type** of `var` and `D` is the **dynamic type** of `var`. If we attempt to access a field or method of `var`, which one is called?

- If `X` is a **field**, the field from the **static type** of `var` will be used.
- If `X` is a **method**, then it depends on whether or not it is static:
  - If `X` is a **static method**, then the method from the **static type** of `var` will be used
  - If `X` is a **non-static method**, then Java will use dynamic method lookup to determine which class's method to call, starting from the lowest class in the hierarchy.

> Important: `D` must be either a child class of or equal to `S`, or it is a compile-time error.

Fields

If we have an object of type `Subclass` that extends `Class`:

```
Subclass s = new Subclass();
```

How can we access...

- A field `x` from `Subclass`?

- A field `x` from `Class`?

Fields

If we have an object of type `Subclass` that extends `Class`:

```java
Subclass s = new Subclass();
```

How can we access...

- A field `x` from `Subclass`?

```java
s.x;
```

- A field `x` from `Class`?

Fields

If we have an object of type `Subclass` that extends `Class`:

```java
Subclass s = new Subclass();
```

How can we access...

- A field `x` from `Subclass`?

```java
s.x;
```

- A field `x` from `Class`?

```java
Class c = s;
c.x;
```

Fields

If we have an object of type `Subclass` that extends `Class`:

```java
Subclass s = new Subclass();
```

How can we access...

- A field `x` from `Subclass`?

```java
s.x;
```

- A field `x` from `Class`?

```java
Class c = s;
c.x;
```

  Alternatively, we can cast our variable:

```java
((Class) s).x;
```

Static Methods

If we have an object of type `Subclass` that extends `Class`:

```
Subclass s = new Subclass();
```

How can we access...

- A static method `f()` from `Subclass` (without calling `Suclass.f()`)?
- A static method `f()` from `Class` (without calling `Class.f()`)?

Static Methods

If we have an object of type `Subclass` that extends `Class`:

```java
Subclass s = new Subclass();
```

How can we access...

- A static method `f()` from `Subclass` (without calling `Suclass.f()`)?

```java
s.f();
```

- A static method `f()` from `Class` (without calling `Class.f()`)?

Static Methods

If we have an object of type `Subclass` that extends `Class`:

```
Subclass s = new Subclass();
```

How can we access...

- A static method `f()` from `Subclass` (without calling `Suclass.f()`)?

```
s.f();
```

- A static method `f()` from `Class` (without calling `Class.f()`)?

```
Class c = s;
c.f();
```

Static Methods

If we have an object of type `Subclass` that extends `Class`:

```
Subclass s = new Subclass();
```

How can we access...

- A static method `f()` from `Subclass` (without calling `Suclass.f()`)?

```
s.f();
```

- A static method `f()` from `Class` (without calling `Class.f()`)?

```
Class c = s;
c.f();
```

  Again, we can simply cast our variable:

```
((Class)s).f();
```

Non-static Methods

If we have an object of type `Subclass` that extends `Class`:

```
Subclass s = new Subclass();
```

How can we access...

- A non-static method `f()` from `Subclass`, assuming that the method is defined in both `Class` and `Subclass`?
- A non-static method `f()` from `Class`, assuming that the method is defined in both `Class` and `Subclass`?

Non-static Methods

If we have an object of type `Subclass` that extends `Class`:

```java
Subclass s = new Subclass();
```

How can we access...

- A non-static method `f()` from `Subclass`, assuming that the method is defined in both `Class` and `Subclass`?

```java
s.f();
```

- A non-static method `f()` from `Class`, assuming that the method is defined in both `Class` and `Subclass`?

Non-static Methods

If we have an object of type `Subclass` that extends `Class`:

```java
Subclass s = new Subclass();
```

How can we access...

- A non-static method `f()` from `Subclass`, assuming that the method is defined in both `Class` and `Subclass`?

```java
s.f();
```

- A non-static method `f()` from `Class`, assuming that the method is defined in both `Class` and `Subclass`?
  This is impossible! This is a feature of Java, not a bug. When you override a non-static method in your parent class, you are specifying a *more specific* action for your subclass to take. If you require the original behaviour of the parent class's method, it is much better design to create another method.

Stack and Heap Diagrams

Draw a picture of what memory looks like when execution reaches the commented line:

```
class Foo {
    int[] x;
    String s;

    public void bar(int x, Foo f) {
        this.x[x] = x;
        f.s = this.s;
        if (f.s != null) {
            // Draw what memory looks like here!
        } else {
            s = "herp derp";
            f.bar(++x, this);
        }
    }

    public static void main(String[] args) {
        Foo f = new Foo();
        f.x = new int[4];
        f.bar(2, f);
    }
}
```

Stack and Heap Diagrams

Solution:

```
+-----------------------+---------------------------------+
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|                       |                                 |
|-----------------------|                                 |
| main                  |                                 |
|        +-+            |                                 |
|   args |?|            |                                 |
|        +-+            |                                 |
+-----------------------+---------------------------------+
          Stack                          Heap
```

Stack and Heap Diagrams

Solution:

```
+-----------------------+----------------------------------+
| bar              +-+  |       +----------+               |
|                x |3|  |       |      +-+ |    +-+-+-+-+  |
|        +-+       +-+  |       |   x  |+------>|0|0|2|3|  |
|  this  |+-------------------->|      +-+ |    +-+-+-+-+  |
|        +-+       +-+  |       |          |               |
|                f |+---------->|      +-+ |               |
|                  +-+  |       |   s  |+------+           |
|-----------------------|       |      +-+ |   |           |
| bar              +-+  |       +----------+   |           |
|                x |3|  |          |  |  |     |           |
|        +-+       +-+  |          |  |  |     |           |
|   this |+------------------------|--|--+     |           |
|        +-+       +-+  |          |  |        |           |
|                f |+--------------|--+        v           |
|                  +-+  |          |     +-------------+   |
|-----------------------|          |     |             |   |
| main                  |          |     | "herp derp" |   |
|        +-+       +-+  |          |     |             |   |
|   args |?|     f |+--------------+     +-------------+   |
|        +-+       +-+  |                                  |
+-----------------------+----------------------------------+
          Stack                         Heap
```

Short Answer

How can you test if Java's implementation of `LinkedList` is singly linked or doubly linked? Assume that you only have one method:

```
Object get(int index)
```

which returns the element at index `index`.

Cyclic Linked Lists

Complete the method for detecting if a singly linked list has a cycle:

```
public static boolean containsCycle(SList myList){
    SListNode a = myList.head;
    SListNode b = myList.head;

    /*
     * Your code goes here. Available SListNode
     * instance variables: next, item.
     */
}
```

Cyclic Linked Lists

Solution:

```java
public static boolean containsCycle(SList myList){
    SListNode a = myList.head;
    SListNode b = myList.head;

    while((a.next != null) && (b.next != null)){
        a = a.next;
        b = b.next;
        if(b.next != null){
            b = b.next;
        }

        if(a == b){
            return true;
        }
    }

    return false;
}
```

Reversing Linked lists

Complete the method for reversing a doubly-linked non-circular tailless linked list

```
public static void reverse(DList myList){
    DListNode b = myList.head;
    DListNode c = myList.head;
    /*
     * Your code goes here. Available DListNode
     * instance variables: next, prev.
     */
}
```

Reversing Linked Lists

Solution:

```
public static void reverse(DList myList){
    DListNode b = myList.head;
    DListNode c = myList.head;

    while(c.next != null){
        c = c.next;
        b.next = b.prev;
        b.prev = c;
        b = c;
    }
    this.head = b;
}
```