

○○
○○○○○○○○○○○○○○○○
○○
○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

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:

```

1 | int i = 0;
2 | while (i < 5) {
3 |     System.out.println();
4 |     i++;
5 | }

```

```

1 | int j = 0;
2 | for (; j < 5; j++) {
3 |     System.out.println();
4 | }

```

```

●○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○○

```

```

○○
○○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○○

```

True/False

The two loops print the same number of lines:

```

1 | int i = 0;
2 | while (i < 5) {
3 |     System.out.println();
4 |     i++;
5 | }

```

```

1 | int j = 0;
2 | for(; j < 5; j++) {
3 |     System.out.println();
4 | }

```

True



True/False

This is a valid Java statement:

```
1 | int [][] [] array = new int [] [] [] ;
```



True/False

This is a valid Java statement:

```
1 | int [][] [] array = new int [] [] [] ;
```

False



True/False

This is a valid Java statement:

```
1 | int [][] [] array = new int [][] [];
```

False

This is a valid Java statement:

```
1 | int [][] [] array = new int [3] [][];
```



True/False

This is a valid Java statement:

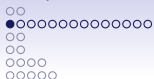
```
1 | int [][] [] array = new int [][] [];
```

False

This is a valid Java statement:

```
1 | int [][] [] array = new int [3] [][];
```

True

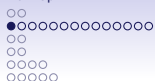


Multiple Choice

What is the value of k at the end?

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

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

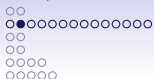


Multiple Choice

What is the value of k at the end?

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

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

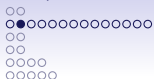


Multiple Choice

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

```
1 public static void main(String[] args) {  
2     String s1 = Hello World ;  
3     String s2 = s1;  
4     String s3 = new String(s1);  
5 }
```

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

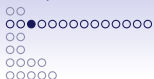


Multiple Choice

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

```
1 public static void main(String[] args) {  
2     String s1 = Hello World ;  
3     String s2 = s1;  
4     String s3 = new String(s1);  
5 }
```

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?

```

1  public static void main(String[] args) {
2      Robot chell = new Robot();
3      String s = "hello human";
4      chell.cake(s);
5      System.out.print(s + " " + chell.s);
6  }
7  public Robot() {
8      String s = "I have a cake";
9      public void cake(String s) {
10         this.s = "cake is a lie";
11         s = "bye human";
12     }
13 }

```

○○
○○○●○○○○○○○○○○
○○
○○
○○○
○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

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"

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?

```

1  public static void main(String[] args) {
2      America Bob = new America();
3      America Mary = new America();
4      Bob.earnMoney(100);
5      Mary.earnMoney(1000);
6      System.out.println(America.publicDebt);
7  }
8  public America() {
9      int myMoney = 0;
10     static int publicDebt = 1000000; //1,000,000
11     public void earnMoney(int wage) {
12         myMoney += wage;
13         publicDebt += wage * 1000;
14     }
15 }
```

○○
○○○○○●○○○○○○○○
○○
○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

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?

```

1  public static void main(String[] args) {
2      myWallet Bob = new myWallet();
3      Bob.earnMoney();
4      if (Bob.hasMoney) {
5          System.out.println("Yay!");
6      } else {
7          System.out.println("Awww");
8      }
9  }
10 public myWallet() {
11     boolean hasMoney = false;
12     public static void earnMoney() {
13         hasMoney = true;
14     }
15 }

```

○○
○○○○○○○●○○○○○
○○
○○
○○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

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?

```
1 | public static void main(String[] args) {  
2 |     int a = 5;  
3 |     int b = 4;  
4 |     int c = a / b;  
5 |     System.out.println(c);  
6 | }
```

○○
○○○○○○○○○●○○○○
○○
○○
○○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

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?

```
1 public static void main(String[] args) {  
2     int a = 5;  
3     double b = 4;  
4     double c = a / b;  
5     System.out.println(c);  
6 }
```


○○
○○○○○○○○○○●○○
○○
○○
○○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

Multiple Choice

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

○○
○○○○○○○○○○●○○
○○
○○
○○
○○○
○○○○

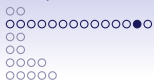
○○
○○○○○○
○
○○○

○○○

○
○○○○

Multiple Choice

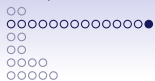
1. 1.25
2. 1
3. 2
4. Error



Multiple Choice

What is printed after the following code is executed?

```
1 | public static void main(String[] args) {  
2 |     double a = 5.5;  
3 |     int b = a;  
4 |     System.out.println(b);  
5 | }
```



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:

```

1  boolean isSorted(int[] array) {
2      // array: array to test if sorted if not
3      // Returns: true if array is sorted in increasing
4      // order
5
6      for (int i = ___; _____; i++)
7          if (array[___] < array[___]) {
8              return false;
9          }
10     return true;
11 }
```

○○
○○○○○○○○○○○○○○○○
●●
○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

○
○○○○

Fill in the blanks

Solution:

```
1 | boolean isSorted(int[] array) {  
2 |     for (int i = 1; i < array.length; i++)  
3 |         if (array[i] < array[i - 1]) {  
4 |             return false;  
5 |         }  
6 |     return true;  
7 | }
```



Fill in the blanks

Fill in the following function:

```

1  | boolean isPalindrome(int[] array) {
2  |     // Returns: true if the array is a palindrome
3  |
4  |     int lower = ___, upper = ___;
5  |
6  |     while (___) {
7  |         if (array[lower] != array[upper]) {
8  |             return false;
9  |         }
10 |         lower++;
11 |         upper--;
12 |     }
13 |
14 |     return true;
15 | }
```




Fill in the blanks

Solution:

```

1  boolean isPalindrome(int[] array) {
2      int lower = 0, upper = array.length - 1;
3
4      while (lower <= upper) {
5          if (array[lower] != array[upper]) {
6              return false;
7          }
8          lower++;
9          upper--;
10     }
11
12     return true;
13 }
```



Fill in the blanks

Fill in the following function:

```

1  boolean hasPalindrome(int[] array, int length) {
2      // Returns: true if array contains a palindrome
3      // of length at least i
4
5      for (int i = 0; i <= ____; i++) {
6          for (int j = ____; j <= ____; j++) {
7              if (isPalindrome(Arrays.copyOfRange(array, i, j))) {
8                  return true;
9              }
10         }
11     }
12
13     return false;
14 }
```

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○●○○
○○○○○

```

```

○○
○○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○○

```

Fill in the blanks

Solution:

```

1  boolean hasPalindrome(int[] array, int length) {
2      for (int i = 0; i <= array.length - length; i++) {
3          for (int j = i + length; j <= array.length; j++) {
4              if (isPalindrome(Arrays.copyOfRange(array, i, j))) {
5                  return true;
6              }
7          }
8      }
9      return false;
10 }

```



Fill in the blanks

Fill in the following function:

```

1  int largestPalindrome(int[] array) {
2      // Returns: the length of the longest
3      // palindrome in the array
4
5      for (int i = ___; ___; ___) {
6          if (____) {
7              return i;
8          }
9      }
10
11     return 0;
12 }
```

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○●
○○○○○

```

```

○○
○○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○○

```

Fill in the blanks

Solution:

```

1  int largestPalindrome(int[] array) {
2      for (int i = array.length; i >= 1; i--) {
3          if (hasPalindrome(array, i)) {
4              return i;
5          }
6      }
7
8      return 0;
9  }

```



What will be printed?

What is printed after the following code is executed?

```

1  public static void main(String[] args) {
2      String s = "Is this the real life?";
3      change(s);
4      System.out.println(s);
5  }
6  public static void change(String s) {
7      s = "Is this just fantasy?";
8  }

```

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?

```

1 public static void main(String[] args) {
2     String s = "Is this the real life?";
3     change(s);
4     System.out.println(s);
5 }
6 public static void change(String s) {
7     s = "Is this just fantasy?";
8 }

```

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?

```
1 public static void main(String[] args) {  
2     int[] arr = {1, 2, 3};  
3     change(arr);  
4     System.out.println(arr[0]);  
5 }  
6 public static void change(int[] i) {  
7     i[0] = 5;  
8     i = null;  
9 }
```

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



What will be printed?

What is printed after the following code is executed?

```
1 public static void main(String[] args) {  
2     int[] arr = {1, 2, 3};  
3     change(arr);  
4     System.out.println(arr[0]);  
5 }  
6 public static void change(int[] i) {  
7     i[0] = 5;  
8     i = null;  
9 }
```

- 1
- 5
- null
- error



What will be printed?

What is printed after the following code is executed?

```

1 | public static void main(String[] args) {
2 |     int herp = 4;
3 |     int derp = 6;
4 |     herp = derp;
5 |     herp = herp + 1;
6 |     System.out.println(derp);
7 | }

```

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



What will be printed?

What is printed after the following code is executed?

```

1 | public static void main(String[] args) {
2 |     int herp = 4;
3 |     int derp = 6;
4 |     herp = derp;
5 |     herp = herp + 1;
6 |     System.out.println(derp);
7 | }
  
```

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



What will be printed?

What is printed after the following code is executed?

```
1 public static void main(String[] args) {  
2     String x = "Caught in a landslide,";  
3     String y = "No escape from reality";  
4     String z = x;  
5     x = y;  
6     System.out.println(z);  
7 }
```

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?

```
1 public static void main(String[] args) {  
2     String x = "Caught in a landslide,";  
3     String y = "No escape from reality";  
4     String z = x;  
5     x = y;  
6     System.out.println(z);  
7 }
```

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?

```
1 | Panda p = new Panda();  
2 | Animal a = p;  
3 | boolean wat = (a == p);  
4 | 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?

```
1 | Panda p = new Panda();  
2 | Animal a = p;  
3 | boolean wat = (a == p);  
4 | 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:

```
1 | class Class {  
2 |     public void foo(int x) { ... }  
3 | }
```

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

```
1 | class Subclass extends Class {  
2 |     public int foo(int y) { ... }  
3 | }
```

1. True
2. False


```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

●○
○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○○

```

Overriding Methods

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

```

1 | class Class {
2 |     public void foo(int x) { ... }
3 | }

```

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

```

1 | class Subclass extends Class {
2 |     public int foo(int y) { ... }
3 | }

```

1. True
2. False

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

○○●
○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○

```

Overriding Methods

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

```

1 | class Class {
2 |     public void foo(Object o) { ... }
3 | }

```

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

```

1 | class Subclass extends Class {
2 |     public void foo(String s) { ... }
3 | }

```

1. True
2. False

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○○

```

```

○○●
○○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○○

```

Overriding Methods

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

```

1 | class Class {
2 |     public void foo(Object o) { ... }
3 | }

```

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

```

1 | class Subclass extends Class {
2 |     public void foo(String s) { ... }
3 | }

```

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.

```
1 | Class c = new Class();  
2 | 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.

```
1 | Class c = new Class();  
2 | 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.

```
1 | Subclass c = new Class();  
2 | 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.

```
1 | Subclass c = new Class();  
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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.

```
1 | Object c = new Class();  
2 | 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.

```
1 | Object c = new Class();  
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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.

```

1 | Class c = new Subclass();
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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.

```
1 | Class c = new Subclass();  
2 | 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:

```

1 | // S and D are predefined classes
2 | S var = new D();
3 | 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:

```
1 | 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:

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

How can we access...

- A field x from Subclass?

```
1 | s.x;
```

- A field x from Class?

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

○○
○○○○○○
○
●○○

```

```

○○○

```

```

○
○○○○

```

Fields

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

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

How can we access...

- A field x from Subclass?

```
1 | s.x;
```

- A field x from Class?

```
1 | Class c = s;
2 | c.x;
```

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

○○
○○○○○○
○
●○○

```

```

○○○

```

```

○
○○○○

```

Fields

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

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

How can we access...

- A field x from Subclass?

```
1 | s.x;
```

- A field x from Class?

```
1 | Class c = s;
2 | c.x;
```

Alternatively, we can cast our variable:

```
1 | ((Class) s).x;
```

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

○○
○○○○○○○
○
○○●○

```

```

○○○

```

```

○
○○○○

```

Static Methods

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

```
1 | 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:

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

How can we access...

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

```
1 | 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:

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

How can we access...

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

```
1 | s.f();
```

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

```
1 | Class c = s;
2 | c.f();
```

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

○○
○○○○○○○
○
○○●○

```

```

○○○

```

```

○
○○○○

```

Static Methods

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

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

How can we access...

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

```
1 | s.f();
```

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

```
1 | Class c = s;
2 | c.f();
```

Again, we can simply cast our variable:

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

```

○○
○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○

```

```

○○
○○○○○○
○
○○●

```

```

○○○

```

```

○
○○○○

```

Non-static Methods

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

```
1 | 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:

```
1 | 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?

```
1 | 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:

```
1 | 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?

```
1 | 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:

```

1  class Foo {
2      int[] x;
3      String s;
4
5      public void bar(int x, Foo f) {
6          this.x[x] = x;
7          f.s = this.s;
8          if (f.s != null) {
9              // Draw what memory looks like here!
10         } else {
11             s = "herp derp";
12             f.bar(++x, this);
13         }
14     }
15
16     public static void main(String[] args) {
17         Foo f = new Foo();
18         f.x = new int[4];
19         f.bar(2, f);
20     }
21 }

```

```

○○
○○○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○○

```

```

○○
○○○○○○○
○
○○○

```

```

○●○

```

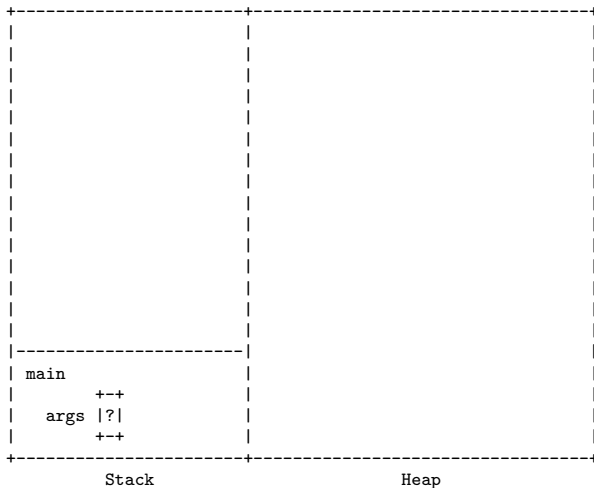
```

○
○○○○

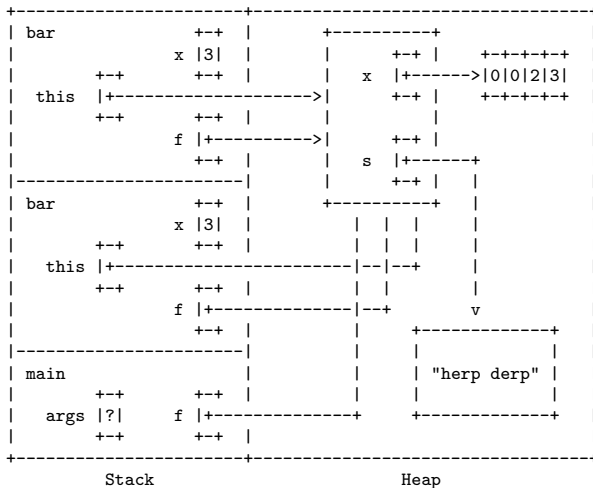
```

Stack and Heap Diagrams

Solution:



Solution:



○○
○○○○○○○○○○○○○○○○
○○
○○
○○
○○○○
○○○○○

○○
○○○○○○○
○
○○○

○○○

●
○○○○

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:

```

1  public static boolean containsCycle(SList myList){
2      SListNode a = myList.head;
3      SListNode b = myList.head;
4
5      /*
6       * Your code goes here. Available SListNode
7       * instance variables: next, item.
8       */
9  }
```



Cyclic Linked Lists

Solution:

```

1  public static boolean containsCycle(SList myList){
2      SListNode a = myList.head;
3      SListNode b = myList.head;
4
5      while((a.next != null) && (b.next != null)){
6          a = a.next;
7          b = b.next;
8          if(b.next != null){
9              b = b.next;
10         }
11
12         if(a == b){
13             return true;
14         }
15     }
16
17     return false;
18 }
```



Reversing Linked lists

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

```

1  public static void reverse(DList myList){
2      DListNode b = myList.head;
3      DListNode c = myList.head;
4      /*
5       * Your code goes here. Available DListNode
6       * instance variables: next, prev.
7       */
8  }
```

```

○○
○○○○○○○○○○○○○○
○○
○○
○○○
○○○○
○○○○○

```

```

○○
○○○○○○○
○
○○○

```

```

○○○

```

```

○
○○○●

```

Reversing Linked Lists

Solution:

```

1  public static void reverse(DList myList){
2      DListNode b = myList.head;
3      DListNode c = myList.head;
4
5      while(c.next != null){
6          c = c.next;
7          b.next = b.prev;
8          b.prev = c;
9          b = c;
10     }
11 }
```