# Prelim 1. Solution

CS 2110, 14 March 2017, 7:30 PM

## 1. Name (1 point)

Write your name and NetID at the top of every page of this exam.

## 2. Short Answer (36 points.)

(a) **5 points.** Below are five expressions. To the right of each, write its value.

1. 'c' == 'b' + 1 \text{true}
2. '5' - '0' \text{5}
3. (double) (double) (int) 5.2 == 5 \text{true}
4. ((Double)(Object)(3.2)).equals(3.2) \text{true}
5. k == 0 || 5/k != 8 \text{[note: k is of type int] true}

(b) **4 points.** Consider classes A and B declared to the right. Will these classes compile? If not, give as many reasons as possible for why they won’t compile.

(1) Since A is abstract, the expression newA() is illegal. (2) Since m() is abstract, class B is illegal because it does not override m.

```java
public abstract class A {
    public abstract void m();
}

public class B extends A {
    public void p() {
        A b= new A();
    }
}
```
(c) 5 points. Write Java code to: Assign array element \( b[h] \) to variable \( k \), but if it throws an ArrayIndexOutOfBoundsException, store 0 in \( k \). Do not use an if-statement, conditional expression, switch statement, or loop. Assume that all variables have already been defined.

```java
try {
    k = b[h];
} catch (ArrayIndexOutOfBoundsException e) {
    k = 0;
}
```

(d) 6 points. Put a check mark before each of the following sentences that is correct and an X before each that is incorrect.

1. An abstract class cannot have a constructor because it cannot be instantiated.  \( \text{false} \)
2. A class can extend only one interface.  \( \text{false} \)
3. Methods in an interface are necessarily abstract, but you can make them public or private.  \( \text{false. They have to be public} \)
4. If a subclass implements an interface, its superclass cannot implement that interface.  \( \text{false} \)
5. A local variable declared with type int is automatically initialized to contain 0.  \( \text{false. local variables are uninitialized} \)
6. Every constructor must start with a call on a super-class constructor.  \( \text{false. It could start with } \texttt{"this(...)"} \)

(e) 12 points. To the right is class CC and its subclass CB. Below is method main of class CC — it belongs in class CC.

Execute a call on method main. Write the value that is printed by each println statement to the right of that println statement.

Printed are the ints \( 3 \ 10 \ 6 \ 4 \ 10 \ 15 \)

```java
public static void main(String[] p) {
    CC a = new CC();
    System.out.println(a.x);
    System.out.println(a.y);
    System.out.println(a.m(a));

    CB b = new CB();
    System.out.println(b.x);
    System.out.println(b.y);
    System.out.println(b.m(b));
}
```

```java
public class CC {
    public int x = 2;
    public int y = 10;

    public CC(int p) { x = p; }
    public CC() { this(3); }

    public int m(CC c) {
        return c instanceof CB ? 5 : 6;
    }
}
```

```java
public class CB extends CC {
    public CB() { super(4); }

    @Override public int m(CC c) {
        return y + super.m(c);
    }
}
```
(f) **4 points.** Suppose you have an abstract class \( A \) and its only components are public abstract methods. You would like a class \( B \) to extend \( A \), but \( B \) already extends a class and it can extend only one. How can you rewrite abstract class \( A \) to solve this problem? Make \( A \) an interface.

### 3. Object-Oriented Programming (33 points)

**3 of 5**

(a) **5 points**

To the right are classes H1 and H2. Method \( p() \) is not overridden in class H2.

Modify class H2 so that a variable will contain the number of times during execution that method \( p() \) is called as a method of any object of class H2 (instead of as an object of class H1 only).

Your modifications should consist of inserting a declaration in class H2 and overriding method \( p() \).

```java
public class H1 {
    public void p() { ... }
}
public class H2 extends H1 {

    // no. times p() called as a component of an object of class H2
    public static int q;
    public void p() {
        super.p(); q = q + 1;
    }
    ...
}
```

(b) **10 points**

Below are two class declarations. Complete the bodies of the constructor and function `toString` in class Outhouse. Be careful; pay attention to access modifiers.

```java
public class Outhouse
    extends Building {

    private int numb; // number of seats
    /** Constructor: instance at address * ad with s seats */
    public Outhouse(String ad, int s) {
        super(ad);
        numb = s;
    }

    /** Return the building’s address, a * space, and number of seats. */
    public String toString() {
        return super.toString() + " "+ numb;
    }
}
```

```java
public class Building {

    private String address;
    /** A building at address ad. */
    public Building(String ad) {
        address = ad;
    }

    /** Return this building’s address */
    public String toString() {
        return address;
    }
}
```
(c) 5 points Complete the body of method equals, which belongs in class Outhouse:

```java
/** Return true iff ob is an Outhouse and * ob has the same number of seats as this Outhouse. */
public @Override boolean equals(Object ob) {
    if (!(ob instanceof Outhouse)) return false;
    return numb == (((Outhouse)ob).numb);
}
```

(d) 5 points Write down the steps in executing a method call m(args).
1. Push a frame for the call onto the call stack.
2. Assign values of arguments to the parameters.
3. Execute the method body.
4. Pop frame for call from call stack; If this is a function push return value onto call stack.

(e) 8 points Consider the interface and class declarations given below. Next to each piece of Java code in the righthand column, write whether it produces no error, a run-time error, or a compile-time error. (Assume that each piece is independent of the others.)

```
interface J1 {}
interface J2 {}
interface J3 extends J1 {}
class D1 implements J2 {}  // (a)
class D2 implements J2 {}  // (b)
class D3 implements J3 {}  // (c)
class D4 extends D2 implements J1 {} // (d)
```

```
(a) J2 a= new J2(); syntax –compiletime  // (a)
(b) J2 b= new D2(); no error  // (b)
(c) D3 c= new D4(); syntax –compiletime  // (c)
(d) D2 d= new D4(); no error  // (d)
```

```
(g) J2 g1= new D2(); no error  // (g)
    D4 g2= new D4(); no error  // (g)
    g2= g1; syntax –compiletime
```

```
(h) J1 h1= new D4(); no error  // (h)
      J2 h2= new D2(); no error  // (h)
      h2= h1; syntax –compiletime
```

4. Recursion (15 Points)

(a) Write the body of recursive function nf, whose specification and header appear below. Do not use loops. Use only recursion.

```java
/* Return the number of times b[k] appears in a row at the beginning of b[k..] * Precondition: 0 <= k < b.length. * Examples: For b containing [2, 2, 2, 3, 2, 6], * nf(b, 0) = 3 and nf(b, 2) = 1. */
public static int nf(int[] b, int k) {
    if (k == b.length-1) return 1;
    if (b[k] != b[k+1]) return 1;
    return 1 + nf(b, k+1); }
```

(b) Below is function putBlank. It is complete except for the base-case if-condition. Circle all possible expressions from the list below that could be used for the base-case if-condition.
1. $s.length() < 2$ no
2. $s.length() <= 2$ yes
3. $s.length() == 2$ no
4. `Integer.parseInt(s) < 100` yes
5. $s.length() == 0$ no

/** Return s formatted by inserting a blank before every second digit.  
* E.g. ”1000” is formatted as ”10 00”, ”56” is ”56”, and ”1234567” is ”1 23 45 67”.  
* Precondition: s is a non-signed integer and the leftmost digit is not 0. **/

public String putBlank(String s) {
  if (base-case if-condition) return s;
  return putBlank(s.substring(0,s.length()-2)) + ' ' + s.substring(s.length()-2);
}

5. **Loop Invariants (15 points)**

(a) 2 points  State the formula for the number of values in array segment $b[h..k-1]$. $k - h$

(b) 13 points  Consider the following precondition, invariant, and postcondition. The postcondition has two alternatives —either section $b[0..h]$ or section $b[j + 1..k]$ is empty (the other one might be, but it is not necessary).

Precondition: $b$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>h</td>
<td>j</td>
<td>n</td>
</tr>
</tbody>
</table>

Invariant: $b$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>h</td>
<td>j</td>
<td>k</td>
</tr>
</tbody>
</table>

Postcondition: $b$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>h</td>
<td>j</td>
<td>k</td>
</tr>
</tbody>
</table>

OR $b$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>h</td>
<td>j</td>
<td>k</td>
</tr>
</tbody>
</table>

Write a loop with initialization that uses the invariant given above to implement the comment given below. Thus, the loop should continue as long as both ? sections are non-empty. Assume that $b$, $j$, and $n$ are already initialized. Identifier $x$ can’t be used in the program; it just stands for the value in $b[j]$. Don’t declare variables, but do assign appropriate values to $h$ and $k$ wherever necessary. To swap $b[i]$ and $b[j]$, just say, ”Swap $b[i]$ and $b[j].” Your grade depends only on how well you use the four loopy questions to write the code.

```java
// Given the Precondition as shown above, swap values of array  
// segment b[0..n] so that the Postcondition holds.
int h= j-1; int k= n;
while (0 <= h && j < k) {
  if (b[h] <= b[j]) h= h-1;
  else {Swap b[h] and b[j]; k= k-1;}
}
```