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. \((\text{int})'b' == 'b'\) true. Remember that char is a number type.

2. \((\text{char})('a' +3)\) 'd'

3. \(\text{new Boolean(false) == new Boolean(false)}\) false. Each new-expression creates a new object, and the pointers to these objects are different.

4. \(((\text{Object})(\text{new Integer(3))).equals(3)\) true

5. \(k != 0 && 5/k == 8\) [note: k is of type int] false. Short-circuit evaluation is used.

(b) 4 points. Consider a class \(C\) with a method \(m\). What are the two consequences of making \(C\) and \(m\) abstract? Making \(C\) abstract means that objects of class \(C\) cannot be created. Making \(m\) abstract means that any non-abstract subclass must override \(m\).

(c) 5 points. Function \(\text{Integer.parseInt(String s)}\) returns the int value of the integer that is in String \(s\). But if \(s\) does not contain an integer, the function throws a \(\text{NumberFormatException}\). Write a statement that stores in variable \(dn\) the value of the function call:

\[ \text{Integer.parseInt(somestring)} \]

but stores 1 in \(dn\) if a \(\text{NumberFormatException}\) is thrown.

```java
try {
    dn= Integer.parseInt(somestring);
} catch (NumberFormatException e) {
    dn= 1;
}
```
(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. A class can extend only one non-abstract class but any number of abstract classes. false
2. All fields in an abstract class must be public. false
3. An abstract class cannot have a constructor because it cannot be instantiated. false
4. If a class implements an interface, its subclasses must not implement that interface. false
5. A local variable declared at the beginning of a method maintains its value from one call of the method to the next. false
6. Every constructor must start with a call on a super-class constructor. false

(e) 12 points. To the right is class M1 and its subclass M2. Below is method main of class M1 —it belongs in class M1.

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

```java
public class M1 {
    public int x = 2;
    public int y = 100;
    public M1() { this.x = 3; }
    public M1(int x) { this.x = x; }
    public int m() {
        return this instanceof M2 ? 5 : 6;
    }
}

public class M2 extends M1 {
    public M2() { super(4); }
    public @Override int m() {
        return 100 + super.m();
    }
}
```

```java
public static void main(String[] p) {
    M1 a = new M1();
    System.out.println(a.x); //ans: 3
    System.out.println(a.y); //ans: 100
    System.out.println(a.m()); //ans: 6

    M2 b = new M2();
    System.out.println(b.x); //ans: 4
    System.out.println(b.y); //ans: 100
    System.out.println(b.m()); //ans: 105
}
```

(f) 4 points. What is the purpose of a constructor? What constructor does Java insert into a class C if no constructor is defined in it? The purpose of a constructor is to initialize fields so that the class invariant is true. If no constructor is defined in class C, Java inserts this one: public C(){} .
3. **Object-Oriented Programming** (33 points)

(a) 5 points
To the right are classes K1 and K2. Method m() is not overridden in class K2.

Modify class K1 so that a variable will contain the number of times during execution that method m() is called as a method of an object of class K2 (instead of as an object of class K1 only).

Your modifications should consist of inserting a declaration in class K1 and adding code at the beginning of method m().

```java
public class K1 {
    // no. times m called in a K2 object
    static int c;

    public void m() {
        if (this instanceof K2) c = c + 1;
        ...
    }
}
```

```java
public class K2 extends K1 { ... }
```

(b) 10 points
Below are two class declarations. Complete the bodies of the constructor and function `toString` in class `Surgeon`. Be careful; pay attention to access modifiers.

```java
public class Doctor {
    private String name;

    /** A doctor named n.
     * Precond.: no space in n */
    public Doctor(String n) {
        name = n;
    }

    /** Return this doctor's name */
    public String toString() {
        return name;
    }
}
```

```java
public class Surgeon extends Doctor {
    private int ops; // no. of operations

    /** Constructor: instance with name n
     * and op operations
     * Precond.: no space in n */
    public Surgeon(String n, int op) {
        super(n);
        ops = op;
    }

    /** Return this surgeon's name, a space, and number of operations. */
    public String toString() {
        return super.toString() + " " + ops;
    }
}
```

(c) 5 points
Complete the body of method `equals`, which belongs in class `Doctor`:

```java
/** Return true iff ob is a Doctor and
 * ob has the same name as this Doctor. */
public @Override boolean equals(Object ob) {
    if (!(ob instanceof Doctor)) return false;
    return name.equals(((Doctor)ob).name);
}
```

(d) 5 points
Write down the steps in evaluating a new-expression `new C(args)`.

1. Create (draw) an instance of class C, with default values for the fields;
2. Execute the constructor call C(args);
3. Return as value of the new-expression the name of (pointer to) the created object.
(e) 8 points
Consider the interface and class declarations given below. Next to each piece of Java code in the right-hand 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.)

Here’s a hint: First draw an object.

interface I1 {...}
interface I2 {...}
interface I3 extends I1 {...}
class C1 implements I1 {...}
class C2 implements I2 {...}
class C3 implements I3 {...}
class C4 extends C3 implements I2 {...}

(a) I2 a = new I2(); // Compile-time error
(b) I2 b = new C2(); // no error
(c) C3 c = new C4(); // no error
(d) C2 d = new C4(); // Compile-time error
(e) C4 e = new C3(); // Compile-time error
(f) C4 f = (C4)(new C3()); // Runtime error
(g) I1 g1 = new C1(); // no error
    C4 g2 = new C4(); // no error
    g1 = g2; // no error
(h) I1 g1 = new C4(); // no error
    I2 g2 = new C2(); // no error
    g2 = g1; // Compile-time error

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. Here is a restriction, which should help you hone in on a simple solution: The only String functions you should use are charAt, length, and substring.

/** Return the number of times the first char of s appears at the beginning of s.
 * Precondition: s is not null and contains at least 1 char.
 * Example: nf("bbbcb$b") = 3.
 * Example: nf("bcb$bbb") = 1. */
public static int nf(String s) {
    if (s.length() == 1) return 1;
    if (s.charAt(0) != s.charAt(1)) return 1;
    return 1 + nf(s.substring(1));
}

(b) Below is function comfy. 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() < 3 no
2. s.length() ≤ 3 yes
3. s.length() == 3 no
4. Integer.parseInt(s) < 1000 yes
5. s.length() == 0 no
/** Return s formatted by adding a comma before every third digit.
 * E.g. 1000 is formatted as 1,000, 56 is 56, and 1234567 is 1,234,567.
 * Precondition s is a non-signed integer and the leftmost digit is not 0. **/
 * public String comfy(String s) {
 * if ( base-case if-condition ) return s;
 * return comfy(s.substring(0,s.length()-3)) + ',' + s.substring(s.length()-3);
 * }

5. Loop Invariants (15 points)

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

\[ k - h \]  // it’s Follower − First

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

<table>
<thead>
<tr>
<th>Precondition: (b)</th>
<th>(0)</th>
<th>(j)</th>
<th>(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>(x)</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>(0)</td>
<td>(h)</td>
<td>(j)</td>
<td>(k)</td>
</tr>
</tbody>
</table>

**Invariant:**

<table>
<thead>
<tr>
<th>Precondition: (b)</th>
<th>(0)</th>
<th>(j)</th>
<th>(k)</th>
<th>(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\leq x)</td>
<td>?</td>
<td>(x)</td>
<td>?</td>
<td>(\geq x)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postcondition: (b)</th>
<th>(0)</th>
<th>(j)</th>
<th>(k)</th>
<th>(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\leq x)</td>
<td>(x)</td>
<td>?</td>
<td>(\geq x)</td>
<td></td>
</tr>
</tbody>
</table>

OR

<table>
<thead>
<tr>
<th>Precondition: (b)</th>
<th>(0)</th>
<th>(j)</th>
<th>(k)</th>
<th>(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\leq x)</td>
<td>?</td>
<td>(x)</td>
<td>(\geq x)</td>
<td></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.

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