Answers are given in bold font.

1. True/False. Please circle the correct answer on this sheet.

   (a) **F** In a class implementing the Runnable interface, the **this** object is the thread itself.
   (b) **T** If class B is a subclass of class A, then B’s constructor can explicitly call A’s constructor using **super**.
   (c) **F** Static overloading refers to the ability to redefine a method of the parent class in the child class.
   (d) **F** Object-oriented design refers to the top-down process of designing the procedures used in the program.
   (e) **T** One can declare a variable of an interface type, but cannot create an object of that type.
   (f) **T** The Class class provides methods for examining the type of an object during execution.
   (g) **F** The private fields of a class are visible within the file in which the class is defined.
   (h) **F** A non-static method cannot refer to any static fields of the class.
   (i) **F** The init() method of an applet is called every time the browser visits the applet.
   (j) **F** Prof. Goldberg is 6’2” tall or less. (He is 6’5” tall, but credit will be given for either answer)

2. Multiple Choice. Circle the letter next to the correct answer. **(The correct answer is underlined)**

   (a) If **public static void main(String[] s)** is defined within class A, then
      i. Since **main** is static, no object of class A can be created within **main**.
      ii. **main** is required to be static since the program cannot create an object of class A before **main** is called.
      iii. The array **s** must be initialized in the constructor for A.
      iv. A compile-time error will result since the string array must be called **args**.
   (b) Given a class T that extends Thread and implements Runnable, the code that would not correctly create a thread and start it executing is:
      i. **new T().start();**
      ii. **new Thread(new T()).start();**
      iii. **new Thread(new T().start());**
      iv. **Thread t = new T(); t.start();**
   (c) A class definition that correctly implements the interface defined by
      ```java
      interface I { boolean plus(I x); }
      ```
      is
      i. **class A implements I { public boolean plus(Obj x) { return true; }}**
      ii. **class A implements I { public boolean plus(I x) { return true; }}**
      iii. **class A implements I { public boolean plus(A x) { return true; }}**
      iv. **class A implements I { public boolean plus(I x) { return x; }}**
   (d) Suppose there is a class T that extends Thread and has a static field **int count** modified within **run()**. Assuming many threads of class T will be running simultaneously, the appropriate synchronization is accomplished by defining T’s **run()** method as:
      i. **public synchronized void run() { ... }**
      ii. **public void run() { synchronized (this) { ... } }**
      iii. **public void run() { synchronized (count) { ... } }**
iv. None of the above

(e) The default visibility for a field of a class is
i. Public
ii. Private
iii. Protected
iv. Package visibility

(f) If class MyException extends Exception, an equivalent way of writing

```
try { f(); } catch(Exception e) { if(e instanceof MyException) { g(e);}}
```

is
i. try { f(); } catch(MyException e) { g(e);}
ii. try { f(); } catch(Exception e) {} catch(MyException e){g(e);}
iii. try { f(); } catch(MyException e) { g(e); } catch (Exception e){}
iv. None of the above

3. Write an applet that, over and over again, draws the string “Hello” moving across the applet from left to right. You can use the method of the Graphics class with the prototype

```java
void drawString(String s, int x, int y)
```

Don’t worry about the font or color of the text.

4. What will be displayed when this program runs?
   In two, count = 3
   In two, count = 9
   Exception ... ClassCastException ...

   ```java
class One {
    int count = 1;
    void f() { System.out.println("In one, count = " + count++); }
}
class Two extends One {
    void f() { count = count * 3;
        System.out.println("In two, count = " + count); }
}
class Three {
    public static void main(String[] args) {
        One x = new Two();
        One y = (One) x;
        x.f();
        y.f();
        Two z = (Two) new One();
        z.f();
    }
}