For each question below answer it and indicate the JLS section used to support the answer. Again, you are only supposed to use the JLS as the reference. Any other source is not accepted.

1a. Define Initialization blocks and provide an example.
b. Define static blocks and provide an example.
c. What is the relationship between constructors and initialization blocks?
d. Can an initialization block replace constructors? which one?
e. Give a programming reason to use initialization blocks.

2. Read carefully on initialization of classes and interfaces and instantiation. in the JLS sections 12.4, 12.5.
You must understand the difference between class initialization and class instantiation. Read carefully on these topics in the JLS before answering the questions.
a. Give 3 different type of examples when a class initialization occurs (i.e., each example illustrate what causes class initialization to start.).
b. Give 2 examples when class instance creation occurs.
c. Does any use of the fields specified as static and final (constant fields) by a client produce the initialization of the class that defines them? Explain.
d. Are referenced variables (in client’s code) initialized with null also used to initialize a class? Explain.
e. Succinctly and in your own words, what is the purpose of class initialization? in your answer state the basic steps taking during initialization.
f. Succinctly and in your own words, what is the purpose of instance creation? in your answer state the basic steps taking during instance creation.
g. Which process must happen first and why: instance creation or class initialization.

h. Consider the following program:

```java
class Super {
    public static final int size = 10;
    public int number;

    Super () { printThree(); }

    void printThree() {System.out.println("three");}
}

class Test extends Super {
    public Test(){
        indiana = texas * factor;
    }

    public static void main(String[] args) {
        Test t = new Test();
        t.printThree();
    }

    void printThree(){
        System.out.println(texas + " and " + indiana + 
            " and " + factor);
    }

    public static final int factor = Super.size * 10;
    private int indiana;
    private int texas = (int)Math.PI;
}
```
It produces the output:
0 and 0 and 100
3 and 300 and 100

Clearly and completely explain the production of such results; notice again, **NOTHING** happens by magic, there is always an agent and a time when things take place; in particular for any instance variable or static variable initialization you must clearly state who caused it and at what time. In your answer while explaining the output you must identify the agent and the time the agent caused something to happen.

3. Read JLS section 8.5 on static initializers. Understand legal initialization of static variables, as well as legal use of static variables not initialized.
   a. Write two programs which produce different output and where the only difference in the programs is the order of declarations of static variables (also called class variables).
   b. Clearly explain the two different results.
   c. What can you conclude about order of declaration of class variables, and methods that may use them?

**Warning**: You will get an automatic ZERO for each question where:
   a. There is a claim that cannot be supported by the JLS.
   b. Typos or poor grammar.

**For graduate students**
Read about class loading and linking in the JLS sections 12.2, 12.3 as well as the paper: Dynamic Class Loading in the Java Virtual Machine by Sheng Liang Gilad Bracha. You can get a copy from the course web reading link.
Also read the java api for ClassLoader.
1. a. Explain the purpose of the class loader.
   b. Can a program use two different class loaders to load classes?
   c. Can a class in the same application be loaded by two different loaders?
2. a. Explain purpose, uses security of system for dynamic class loading.
   b. Clearly describe an application of dynamic class loading.
3. a. Give a review of the steps for class linking and the reason behind each of them.
   b. Discuss problems and solutions of safe class linkage on the face of dynamic class loading.