Full Name: ______________________________________

Please answer each question only within the space provided for each question. All questions have equal points.

1. A good design should have high ________ and low _________. Fill in the blanks and explain what that means and why should be care for that.

2. What are some of the characteristics of a simple design?
3. “Inheritance is required to realize polymorphism.” Discuss in favor or against this statement.

4. Draw a UML diagram for the following relationship: An computer may have any number of peripherals attached to it. A peripheral may be an Printer, a Scanner, a CDWriter, etc. The peripheral may be attached to different types of computers, a PC, a Mac, etc. Each computer has a memory unit and may also have one or more monitors attached to it.
5. Given
   public class Util {
       public int doubleValue(int value) { return value * 2; }
   }

   public class EvenUtil extends Util {
       public int doubleValue(int value) {
           if(isNotEven(value)) throw new RuntimeException("No supported");
           return value * 2;
       }
   }

   Identify the design smells in the above code. For each design smell you identify, explain it and discuss the consequence of that design smell.

6. What is an invariant of a class? Given an example. What OO feature helps to preserve the invariant of a class?
7. Discuss in favor or against the following statement and provide an example: “Runtime Type Identification (RTTI) will always result in violating the Open-Closed Principle.”

8. What are multimethods? How do they differ from polymorphism? How can they contribute to better extensibility? Give an example.
9. Mention five reasons why long methods are bad. Explain.

10. A team with a lot of experience with statically typed languages is interested in starting a project where they would predominantly use a dynamically typed language.

   a. Discuss how this would fundamentally change the way they would design their software.
b. The flexibility of the dynamic languages are great, but they often lack compiler support. What do you recommend the team do to take advantage of the benefits without begin burned by the lax compiler support.