CSSE 220 – Object-Oriented Software Development
Rose-Hulman Institute of Technology

Worksheet 07

Name (Print):_____ Section:_____

1. **Check Your Understanding: Instance, Static, Final**.

Field Type	**What it means**
Instance field	
`final` instance field	
`static` field	
`static final` field	

2. **True or False**

   Every object gets its own copy of an instance field. (T / F)
   A `final` instance field can be changed after the constructor finishes. (T / F)
   A `static` field belongs to the class (T / F)
   A `static final` field is best described as a true **constant** shared by all objects (T / F)

3. **Quest Class**

```
public class Quest {
  private String title;
  private final int xpReward;
  private static int totalQuests;
  private static final int MAX_XP = 500;
}
```

   (a) Each `Quest` has its own **(title / xpRewards, totalQuests, MAX_XP)**
   (b) `xpReward` can be assigned: **(once / many times)**
   (c) `totalQuests` is shared by: **(each object / the class)**
   (d) `MAX_XP` can: **(change / never change)**

4. Review: **Principle 1 Proper Functionality**. This principle answers:

   _____ ?

5. Review: **Principle 2 Design around data**. This principle answers:

   _____ ?

6. **Encapsulation**

   Encapsulation means an object _____ its own data by _____ how it is accessed and modified

7. **Design**. There are two major goals of all program design:

   (a) Easy to _____

   (b) Easy to _____

8. **The Big Picture: OOP Pillars**

   - Encapsulation → objects _____ their data
   - Abstraction → users see a _____ interface
   - Inheritance → related classes _____ structure
   - Polymorphism → one method call, _____ behavior

9. **Pizza Design**. From the problem statement:

   Primary nouns (classes): _____

   Attributes of Order: _____

   Attributes of Pizza: _____

   Verbs that became methods: _____

10. **Bad Design Example**. Consider the Pizza class below which violates Principle 3:

```
1  public class Pizza {
2      private ArrayList<Topping> toppings;
3      public Pizza() {
4          toppings = new ArrayList<>();
5      }
6      public void addTopping(Topping t) {
7          toppings.add(t);
8      }
9      public ArrayList<Topping> getToppings() {
10         return toppings;
11     }
12 }
```

    Why? _____

11. **Before You Leave**

    Write one question you still have about *static, final, or overriding.*