The City College of New York
Department of Compute Science – CSc 221: Software Design Laboratory

Name:_____ ID:_____

Q1) (10p) Briefly define Java Exceptions. When are they used and what is the difference between checked and unchecked exception?

- An indication of a problem has occurred during a program's execution.
- With exception handling, a program can continue executing
- Root class of hierarchy is Exception. Subclasses are error-specific classes for better error management.
- Checked Ex:
  - Compiler enforces a catch-or-declare requirement
  - Subclass of exception by not RuntimeException
- Unchecked Ex: Compiler doesn't enforce them
  - Inherits from RuntimeException, Caused by defects

Q2) (10p) What is the difference between static and non-static methods. Explain the allowed interactions between static and non-static methods and why?

- Static methods are accessible without an instance
- Static methods can access static methods only since non-static methods are only available when an instance is created.
- Non-static methods can access both types

Q3) (10p) Java has two types of data, reference and primitive, explain the difference(s) between the two types?

- Reference types are pointers to objects
- Primitive types are not pointers – data stored directly in their memory location
- Reference types are passed by reference only
- Primitive types are passed by value only

Q4) Complete the missing parts of the class shown in the class diagram. Refer to the section's comments for details.

```
Public class Property {
// a. (5p) define the class' variables
private String name;
private double area;
private ArrayList<String> owners;
```

Property
- name: String
- area: double
- owners: ArrayList<String>
+ Property(String, double)
+ getName() : String
+ getArea() : double
+ setArea(double)
+ addOwner(String)
+ getOwners():ArrayList<String>

```
// b. (10p) define the non-default constructor. Initialize members accordingly
public Property(String name, double area) {
    this.name = name;
    this.area = area;
    this.owners = new ArrayList();
}
```

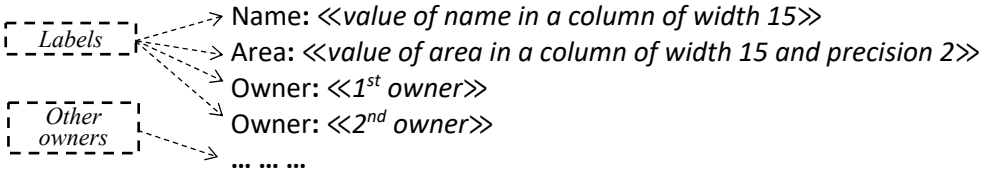
Name:_____ ID:_____

```
// c. (5p) define the accessor (i.e. getter) method for the variable area
public double getArea() {
    return area;
}
```

```
// d. (10p) define the method addOwner. The method will search the ArrayList for a matching value. If the ArrayList
//          does NOT contain the parameter value, it is added to the list.
public void addOwner(String owner) {
    if(!owners.contains(owner))
        owners.add(owner);
}
```

Q5) (40p) Write the main method and complete the following steps:

a. **Declare an instance of the class *Property*. Set *name* to *Pinnacle* and *area* to *700.133***
b. **Using the scanner class, prompt the user to enter a *String* value (spaces not allowed). Add this value to the *owners* ArrayList member of the class.**
c. **Using the *printf* method ONLY, print the instances information as follows:**




```
public static void main(String args[]) { // 5p

    Property prop = new Property("Pinnacle", 700.133); // 7p

    Scanner scanner = new Scanner(System.in); // 3p

    String owner = scanner.next(); // 2p

    scanner.close(); // 2p

    prop.addOwner(owner); // 3p

    System.out.printf("Name: %15s%n", prop.getName()); // 4p

    System.out.printf("Area: %15.2f%n", prop.getArea()); // 5p

    for(String own : prop.getOwners()) // 5p

        System.out.printf("Owner: %s%n", own); // 4p

}
```