1. **Question:** Provide short answers (a few sentences) to each of the following questions:

(a) What does it mean to designate a datum or method as **public** or **private**? Why should some such members be designated **private**?

(b) Why must a recursive method contain a **base case**?

(c) Why must a **constructor** have no return type?

**Answer:**

(a) A member that is **public** is accessible by code from any class/object, while one that is **private** is accessible only from code within that member’s class. Some members should be made **private** if their access or use exposes elements of how the class/object works that does not need to be available to code outside the class. For example, data members should be **private** to avoid non-member code from corrupting those members’ values.

(b) Without a **base case** a recursive method will endlessly call itself. In principle, it execute forever; in practice, the activation stack grows too large and exhausts its allowed memory.

(c) A constructor is only called via the **new** operator when an object is initially created. Since the **new** operator must itself return a pointer to the newly created object, a return value from a constructor could not also be returned.

**Discussion:** [To be added.]
2. **Question:** Consider writing a method that does the following:

- Prompt the user to enter an integer between the min and max values, inclusive.
- Obtain the user’s input as a String by using the following (assumedly) already written method:
  ```java
  public static String getTypedInput()
  ```
- Attempt to convert the obtained input into an int by calling the following Integer class method:
  ```java
  public static int parseInt (String s) throws NumberFormatException
  ```
- If the user’s input is not convertable to an integer, or if the value entered is outside of the range specified by min and max, prompt the user again until this condition is fulfilled.
- Return the converted int.

**Complete this method:**
```
public static int getIntInRange (int min, int max) {
  while (true) {
    System.out.print("Enter a value between " + min + " and " + max + ": ");
    String s = keyboard.nextLine();
    int i;
    try {
      i = Integer.parseInt(s);
    } catch (NumberFormatException e) {
      continue;
    }
    if ((min <= i) && (i <= max)) {
      return i;
    }
  }
}
```

**Answer:**
```
public static int getIntInRange (int min, int max) {
  while (true) {
    System.out.print("Enter a value between " + min + " and " + max + ": ");
    String s = keyboard.nextLine();
    int i;
    try {
      i = Integer.parseInt(s);
    } catch (NumberFormatException e) {
      continue;
    }
    if ((min <= i) && (i <= max)) {
      return i;
    }
  }
}
```

**Discussion:** [To be added.]
3. **Question:** Consider the following two object classes . . .

```java
public class Alpha {

    protected int _x;

    public Alpha (int x) {
        _x = x;
    }

    public void show1 () {
        System.out.println("Alpha 1: "+_x);
    }

    public static void show2 () {
        System.out.println("Alpha 2");
    }

    public void show3 () {
        this.show2();
    }
}

public class Beta extends Alpha {

    private int _x;

    public Beta (int x) {
        super(x);
        _x = x * 2;
    }

    public void show1 () {
        System.out.println("Beta 1: "+_x +" "+super._x);
    }

    public static void show2 () {
        System.out.println("Beta 2");
    }

    public void show3 () {
        this.show2();
    }
}
```
...as well as this static class ...

```java
public class Go {
    public static void main (String[] args) {
        Alpha a = new Beta(4);
        a.show1();
        a.show2();
        a.show3();
    }
}
```

Show the output generated when this program is run by invoking:

```
$ java Go
```

**Answer:**

```
$ java Go
Beta 1: 8 4
Alpha 2
Beta 2
```

**Discussion:** [To be added.]

---

1You may provide short explanations of why you chose that particular output.
4. **Question:** Consider the following recursive method:

```java
public static void doit (int n, char prefix) {
    if (n > 0) {
        System.out.println("a: " + prefix + n);
        doit(n-1, '$');
        System.out.println("b: " + prefix + n);
        doit(n-1, '%');
        System.out.println("c: " + prefix + n);
    }
}
```

Show the output generated when this method is called like so:

```java
doit(3, '!' );
```

**Answer:**

a: !3  
a: $2  
a: $1  
b: $1  
c: $1  
b: $2  
a: %1  
b: %1  
c: %1  
c: $2  
b: !3  
a: %2  
a: $1  
b: $1  
c: $1  
b: %1  
a: %2

\(^2\)Again, explanations, diagrams, or any other demonstration of your thinking is welcome.
b: %1

c: %1

c: %2

c: !

**Discussion:** [To be added.]