Ans Key – Sess 1 – Mar 2024 – MCA – SEM II (BS)

a) State True or False: A Boolean value can be cast to an int.
   (Explain with proper reason)
   False : **We** cannot **cast a boolean to an int in Java**. **We** must use an **if-** statement, or a ternary, to **convert**. A separate method **can** be used to encapsulate and name this logic.
b) Question: True or False: Aggregate operations are mutative operations that modify the underlying collection.

   Answer: False. Aggregate operations do not mutate the underlying collection. In fact, you must be careful to never mutate a collection while invoking its aggregate operations. Doing so could lead to concurrency problems should the stream be changed to a parallel stream at some point in the future.
c) A software object's behavior is exposed through ___.
   Ans : methods

   D interface

   Dynamic method dispatch

   Method can be called dynamically in Java. Whenever, a method is called on an object reference, the declared type of the object reference is checked at compile time to make sure that the method exists in the declared class. At run time, the object reference could be referring to an instance of some subclass of the declared reference type.

   a. It has a method implementation in it. Only default and static methods have implementations.
   b.

      What will be the output of the following program?

```
public class SomethingIsWrong {
   public static void main(String[] args) {
        Rectangle myRect;
        myRect.width = 40;
        myRect.height = 50;
        System.out.println("myRect's area is " +
myRect.area());
    }
}
```

   a. It will generate NullPointerException
   b. myRect's area is 20000
   c. Compiler generates an error
   d. None of the above

Consider the following class:

```
public class IdentifyMyParts {
    public static int x = 7;
    public int y = 3;
}
```

a. **Question**: What are the class variables?

Answer: x

b. **Question**: What are the instance variables?

Answer: y

c. **Question**: What is the output from the following code:

```
IdentifyMyParts a = new IdentifyMyParts();
IdentifyMyParts b = new IdentifyMyParts();
a.y = 5;
b.y = 6;
a.x = 1;
b.x = 2;
System.out.println("a.y = " + a.y);
System.out.println("b.y = " + b.y);
System.out.println("a.x = " + a.x);
System.out.println("b.x = " + b.x);
System.out.println("IdentifyMyParts.x = " + IdentifyMyParts.x);
```

Answer: Here is the output:

```
    a.y = 5
    b.y = 6
    a.x = 2
    b.x = 2
    IdentifyMyParts.x = 2
```

Because `x` is defined as a `public static int` in the class `IdentifyMyParts`, every reference to `x` will have the value that was last assigned because `x` is a static variable (and therefore a class variable) shared across all instances of the class. That is, there is only one `x`: when the value of `x` changes in any instance it affects the value of `x` for all instances of `IdentifyMyParts`.

e. The term "instance variable" is another name for ?
f. a. **non-static field.**
g. b. static field
h. c. class variable
i. d. parameter