True / False

1. (T) Immutable Objects are automatically thread safe. So you don't need any synchronization.
2. (F) You can declare a generic method in a non generic class but you cannot declare a generic interface without declaring a generic method in it.
3. (F) Since Object is a supertype of all reference types in Java, so we can write code like this:

```
Unit<String> us = new Unit<>();
Unit<Object> uo = us;
```

4. (F) Both Adder and SmartAdder methods below are functional interface because both of them contain only one abstract method and the abstract method is not the method defined in Object class.

```
public interface Adder {
    int add (int a, int b);
}
public interface SmartAdder extends Adder {
    int add (double a, double b);
}
```

5. (F) The following code will compile without problems since Java only support single inheritance for class and Object is the super-type of all classes even though we omit "extends Object" most of the time.

```
public enun Size extends Object {
    SMALL, MEDIUM, LARGE }
```

6. (F) In Java 8, Stream's limit() is a terminal operation because it will return a stream of a limited size.
7. (F) If two objects have the same hashcode, the equals() must return true.
8. (F) This is a correct lambda expression: ()-> return "Tricky Exam".
9. (T) The following statements can compile without errors. CheckingAccount is a subclass of Account.

```
List<? extends CheckingAccount> checkingAccounts= new ArrayList<>();
List<? extends Account> accounts = checkingAccounts;
```

10. (F) The following lambda expression is correct.

```
Object o = () -> { System.out.println("Tricky Final Exam!"); );
```

11. (T) A subtype of Enum can be defined so that it is a subtype of java lang comparable.
12. (F) Assume Employee is an existing class with proper attributes and setter/getter methods.
    The following EmployeeInfo class will produce a compiler error.

```
public class EmployeeInfo {
    public void sort(List<Employee> emps, final Employee e) {
        Collections.sort(emps, (e1, e2) -> {
            e.setSalary(10000);
            return e1.getName().compareToIgnoreCase(e2.getName());
        });
    }
}
```

13. (F) The lambda below contains no free variables.

```
(String str) -> System.out.printIn(str);
```

14. (F) SmartAdder is a functional interface.

```
public interface Adder {
    int add (int a, int b);
}
public interface SmartAdder extends Adder {
    int add (double a, double b);
}
```

15. (F) In Java 8, distinct() method is an intermediate and stateless operation in the Stream API.
16. (F) Java Generics provide stronger type checking at runtime.
17. (T) The following statements can compile without errors.

```
List<? extends Integer> intList = new ArrayList<>();
List<? extends Number> numlist = intlist;
```