Makeup Exam(2) – 15 minutes – 20 T/F problems

Submission link: https://canvas.ust.hk/courses/42336/assignments/212224

1. For the following program, if call to test succeeds without any Exception, it will print "apple Fruit Meat " in order. (True/False)

```
try {
    test();
    print("apple ");
} catch(IOException e1){
    System.out.print("Orange ");
} catch(Exception e2){
    System.out.print("Pineapple ");
} finally {
    System.out.print("Fruit ");
}
System.out.println("Meat ");
```

2. The following method is a legal one. (True/False)

```
void foo() throws IOException {//empty
}
```

3 Exactly one of the following programs is incorrect. (True/False)

```
public static void main(String args[]) {
System.out.println("1".toCharArray() instanceof Object);}
```

```
public static void main(String args[]) {
System.out.println(args.length instanceof Object);}
```

4. For the following classes, a call to "new B().a()" will print "CB". (True/False)

```
class A{
      void a() { this.b();}
      void b() { System.out.print("B");}
      }
```

```
class B extends A {
      void b() { System.out.print("C"); super.a();}
}
```

5. For the following classes,

```
class X {
    void foo() {System.out.print("B");}
    static void a() { System.out.print("B");}
      X(){ a(); foo(); }
}
```

```
class Y extends X {
    void foo() {System.out.print("A");}
    static void a()          { System.out.print("A");}
    Y(){ a(); }
}
```

A call to "new Y()" prints "BBA". (True/False)

6. For the following classes,

```
class A{
      private void a() { System.out.print("A");}
       A(){ a(); }
}
```

```
class B extends A {
   private void a(){ System.out.print("B");}
}
```

A call to "new B()" prints "B". (True/False)

7. A method of a subclass can't access the protected members of its parent class. (True/False)

8. In java, programmer needs to explicitly free the allocated memory for objects to avoid memory leak (True/False)

9. Student is a user defined class that contains an integer field score. For the code "Student s; System.out.println(s);" to print out the value of s.score, we need to override the toString method in Student (True/False).

10. For the following classes, A call to "new B()" prints BCAD (True/False)

```
class A{
        { System.out.print("A"); }
      static { System.out.print("B"); }
}
```

```
class B extends A {
      static { System.out.print("C"); }
      B(){ System.out.print("D"); }
}
```

11. In the following program, the output will contain 4 "!". (True/False).

```
class Stack<T>{static {System.out.println("!");}}
class Fruit {static {System.out.println("!");}}
class Apple extends Fruit{
  static {System.out.println("!");}
  public static void main(String[] args) {
    Stack<? extends Fruit> st1 = new Stack<Apple>();
    Stack<Fruit> st2 = new Stack<Fruit>();
  }
}
```

12. Is the right side the correct type-erasure of the left? (True/False)

```
class Parent<T extends String> {
    T content;
    void store(T t) {content = t;}
    T get(){return content;}
}
```

```
class Parent {
    Object content;
    void store(Object t) {content = t;}
    Object get(){return content;}
}
```

13. The UNKNOWN here can ONLY be filled with **Object**. And the output of main function is true. (True/False)

```
class N{}
class M extends N {
    public static void doit(ArrayList<? extends N> list) { // use generics here!
        UNKNOWN el = list.get(0);
        System.out.println(el instanceof M);
    }
    public static void main(String[] args) {
        ArrayList<M> list = new ArrayList<M>();
        list.add(new M());
        M.doit(list);
    }
}
```

14. The following program prints 2 (True/False)

```
class T {
   void m(Integer e) { //print 1}
}
class X<E> extends T {
  void m(E o) { //print 2 }
}
//main
X t = new X<Integer>();
t.m((Object)new Integer(1));
```

15. We learned the following functional interfaces in class:

A. Predicate<T>: *T -> Boolean,* **B.** Function<T, R>: *T -> R,* **C:** Consumer<T>: *T -> void,* **D:** Supplier<T>: *() -> T*

For the following program, the UNKNOWN can be supported by C. (True/False)

```
void func (){
    Map<Integer, UNKNOWN> map = new HashMap<>();
    Integer n = 1;
    map.put(n, n::toString());
}
```

16. A synchronized instance method of class A will try to acquire a lock on the class A. (True/False)

17. We learned the following functional interfaces in class:

A. Predicate<T>: *T -> Boolean,* **B.** Function<T, R>: *T -> R,* **C:** Consumer<T>: *T -> void,* **D:** Supplier<T>: *() -> T*

The UNKNOWN in the following should be "N, M". (True/False)

```
void func (Function<UNKNOWN> foo, Consumer<M> bar, Supplier<N> baz){
        bar.accept(foo.apply(baz.get()))
}
```

18. Consider the following program that uses **synchronized**. The output result will be like either "AA…AABB…BB" or "BB…BBAA…AA" with no interleaving between "A" and "B" (True/False)

```
class Task implements Runnable{
  String name;
  Task(String name) {
    this.name = name;
  }

  public void run() {
    synchronized(this) {    // syncronized here
      for(int i=0; i<10000; i++) {    // a loop to print its name
        System.out.print(name);
      }
    }
  }

  public static void main(String[] args) {
    Task task1 = new Task("A");
    Task task2 = new Task("B");
    Thread t1 = new Thread(task1);
    Thread t2 = new Thread(task2);
    t1.start();
    t2.start();

  }
}
```

19. Any interface that contains exactly 1 method declaration is a functional interface. (True/False)

20. In the following program, the unknown_func() has 2 different possible outputs. (True/False)

```
Class A {
int num = 0;

void unknown_func() {
    Thread t1 = new Thread( ()-> {
        synchronized (this) {
        num += 50;
        }
    });
    Thread t2 = new Thread( ()-> {
        synchronized (this) {
        num += 50;
        }
    });
    t1.start();
    t2.start();
    System.out.print(num);
}
}
```