NATIONAL UNIVERSITY OF SINGAPORE

School of Computing

MAKE UP MID-TERM ASSESSMENT
AY2018/19 Semester 4

CS2030 — Programming Methodology II

17 July 2019 Time Allowed: **1 hour**

INSTRUCTIONS

1. This question paper contains **NINE (9)** questions and comprises **SIX (6)** printed pages.
2. The Answer Sheet contains **TWO (2)** printed pages.
3. Write your **Student Number** and **Tutorial Group** Number with a PEN.
4. Answer **ALL questions** within the space provided on the Answer Sheet.
5. Return only the Answer Sheet, you may keep the question paper.
6. You may write your answers in pencil (at least 2B).
7. You must **write legibly** or marks may be deducted.
8. This is an **OPEN Book** test.
9. Maximum score of this test is **20 marks**.

——— END OF INSTRUCTIONS ———

Multiple Choice Questions **[5 Marks]**

1. Consider the interfaces and classes below.

```
interface A {}
interface B {}
interface C extends A {}
interface D extends A,B {}
interface E extends C,D {}
class F implements B {}
class G extends F implements E {}
class H extends G {}
```

Further consider the following method class `Pair<T,U>` used in class. If we declared a variable `Pair<? super G, ? extends B> pair`. Which of the following instantiation will NOT cause compile-error?

A. `pair = new Pair<H,D>(new F(), new G());`
B. `pair = new Pair<G,D>(new G(), new E());`
C. `pair = new Pair<F,D>(new F(), new G());`
D. There are more than one answer from above.
E. None of the above.

2. Which of the following statement about inheritance in Java is `True`? We are not considering the use of extends in generic (i.e., `<A extends B>`).

A. In `A extends B`, `A` must be a class or an abstract class.
B. In `A extends B`, `B` must be a class or an abstract class.
C. In `A implements B`, `A` must be a class or an abstract class.
D. In `A implements B`, `B` must be a class or an abstract class.
E. None of the above.

Question 3 and 4 uses the code below.

```
class Exceptional {
  public static void f(int i) throws E1 {
    try {
      if(i < -1) { throw new E1(); }
      g(i+1);
    } catch(E2 e) { System.out.println("A"); }
  }
  public static void g(int i) throws E1 {
    try {
      if(i > 0) { throw new E2(); }
      System.out.println("B");
    } catch(E1 e) {
      h(i*i);
    }
```

```
  }
  public static void h(int i) throws E1 {
    try {
      if(i == 1) { throw new E1(); }
      System.out.println("C");
    } catch(Exception e) { }
  }
}
class E1 extends Exception {}
class E2 extends E1 {}
```

3. Which of the following will cause nothing to be printed and let the program runs without any uncaught exception?
   A. `Exceptional.f(-1)`
   B. `Exceptional.f(0)`
   C. `Exceptional.f(1)`
   D. `Exceptional.f(2)`
   E. None of the above.

4. Which of the following will be printed when `Exceptional.f(-2)` is invoked?
   A. `A`
   B. `B`
   C. `C`
   D. More than one character.
   E. None of the above.

5. Consider the class with the following `hashCode` and `equals` method. Note that the attributes, other methods, as well as other classes are omitted.

```
class Hash {
  :
  @Override
  public boolean equals(Object obj) {
    if(obj == this) { return true; }
    if(obj.hashCode() == this.hashCode()) { return true;  }
    if(obj.hashCode() != this.hashCode()) { return false; }
  }
  @Override
  public int hashCode() {
    return -1;
  }
}
```

Which of the following statement is `false` about the class above.

A. No instance of class `Hash` can be equal to any instance of another class using equals method.
B. Any instance of class `Hash` will be equal to any other instance of class `Hash` using equals method.
C. An instance of subclass of `Hash` may be not equal to an instance of class `Hash`.
D. There are no `false` statement in A, B, or C.
E. There are more than one `false` statement in A, B, or C.

Short Answers **[8 marks]**

6. Consider the interfaces and classes below.

```
class A {
  public void f(int x) {
    System.out.println("A");
  }
}
class B extends A {
  public void f(double x) {
    System.out.println("B");
  }
}
class C extends B {
  public void f(int y) {
    System.out.println("C");
  }
}
```

A. What will be printed when `new B().f(1);` is invoked?
B. Write **two (2)** statements involving only two of the three classes above will print "`B`"?

7. Consider the following generic class.

```
class Gen<T> {
  private Object[] elems;
  private int idx;
  public Gen() {
    elems = new Object[100];
  }
  public void add(T elems) {
    elems[idx] = elems;
    idx = idx + 1;
  }
  public void add(Object obj) {
    elems[idx] = obj;
  }
}
```

The code above will have compile-error. Explains **two (2)** errors from the code and describe modification to the code above to remove the compile-error.

8. Consider the classes below.

```
class A {
  private int x;
  public A(int x) {
    this.x = x;
  }
  public A copy() {
    return new A(x);
  }
  public void setX(int x) {
    this.x = x;
  }
  public int getX() {
    return x;
  }
}
class B {
  private int x;
  private A a;
  public B(int x, A a) {
    this.x = x;
    this.a = a;
  }
  public B copy() {
    return new B(x, a);
  }
  public void setX(int x) {
    this.x = x;
    this.a.setX(x);
  }
  public int getX() {
    return this.x + this.a.getX();
  }
}
```

Consider the following code fragment.

```
A a = new A(2);
B b1 = new B(2, a);
B b2 = b1.copy();
b2.setX(3);
System.out.println(b1.getX());
```

A. What is the output of the code above? 5

B. What do we need to change from class `B` such that the output of the code above is `4`? If there is nothing to change, simply write "`Nothing to change`". If there is any changes, make sure that the changes is only in one of the function, and it replaces only a single statement above with another single statement.

Long Answers **[7 Marks]**

9. There are several ways codes can be reused in OOP. Some of the possibilities include inheritance, generic, etc. However, we will not consider the use of `Function` class and/or other materials discussed in Lecture 7 onwards.
   You are given the following code where the implementation for sort is omitted.

```
class DescendingOrder {
  public DescendingOrder() {}
  public final void descending(Integer[] arr) {
    for(int i=0; i<arr.length; i++) {
      for(int j=0; j<arr.length-1; j++) {
        if(arr[i].compareTo(arr[j]) > 0) {
          swap(arr, i, j);
        }
      }
    }
  }
  public final void swap(Object[] arr, int i, int j) {
    Object temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
  }
}
```

   Instead of just sorting Integer array, you want to sort any kind of class that implements `Comparable` interface

   A. Change the method signature of `descending` such that it can accept any class that implements the `Comparable` interface. It should not accept other classes that does not implement the `Comparable` interface. You should not change the first two keywords of `descending` as well as the return type of `descending`.

   Now that you have one function to sort in descending order that accepts multiple types, you want to extend its capability to sort in ascending order. However, you are faced with the following restrictions.

   i. You must implement a class `AscendingOrder` that extends on `DescendingOrder`.
   ii. You must reuse the `descending` method in `DescendingOrder` to avoid code duplication.
   iii. Besides the method descending through restriction (ii) above, you can only use one other method which is `swap` method in `DescendingOrder`.
   iv. You can only use `swap` method as many times as there are elements in the array `arr`.
   v. Your method must also accept any class that `descending` can accept.
   vi. You cannot create a temporary array within `ascending`.

   B. Write the class `AscendingOrder` that contains only a single method `ascending` that satisfies the restriction above.