NATIONAL UNIVERSITY OF SINGAPORE

School of Computing

MID-TERM Assessment
AY2018/19 Semester 4

CS2030 — Programming Methodology II

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

INSTRUCTIONS

1. This question paper contains **TEN (10)** questions and comprises **SEVEN (7)** printed pages.
2. Write your **Student Number** and **Tutorial Group** Number with a **PEN**.
3. Answer **ALL questions** within the answer sheet provided.
4. You may write your answers in pencil (at least 2B).
5. You must **write legibly** or marks may be deducted.
6. This is an **OPEN Book** test.
7. Maximum score of this test is **20 marks**.

——— END OF INSTRUCTIONS ———

Multiple Choice Question **[5 Marks]**

1. Consider the code fragment below.

```
public static double foo(Double x, Double y) {
  return x + y;
}
public static double foo(double x, Double y) {
  return x - y;
}
public static double foo(Double x, double y) {
  return x * y;
}
```

   Consider executing the function call `foo(7.0, 3.0)`. What is the return value?

   A. 10.0

   B. 4.0

   C. 21.0

   D. Compile error

   E. Runtime error

2. Consider the code fragment below.

```
class A {
  public void foo() {
    System.out.println(“A.f”);
  }
}
class B extends A {
  public void foo(int x) {
    System.out.println(“B.f”);
  }
}
```

   Consider the concepts below.

   i. Inheritance
   ii. Overriding
   iii. Overloading

   Which of the concepts above are illustrated in the code fragment above?

   A. (ii) only

   B. (iii) only

   C. (i) and (ii) only

   D. (i) and (iii) only

   E. (i), (ii), and (iii)

3. Consider the interfaces and abstract class below.

```
interface I1 {
  public void f();
  public void h();
}
interface I2 {
  public void f();
  public void g();
}
abstract class AC {
  public abstract void f(int x);
  public void f() {
  }
}
```

Consider the concrete class `C` that extends on `AC` using the following class declaration: `class C extends AC`. With respect to the functions below:

```
   i.   f()
  ii.   g()
 iii.   h()
  iv.   f(int x)
```

How many of the functions above need to be implemented in `C`?

A. 0
B. 1
C. 2
D. 3
E. 4

4. Consider the code fragment below.

```
public static void f() {
  try {
    System.out.println(1);
    throw new Exception();
    System.out.println(2);
  } catch(Exc1 e) {
    System.out.println(3);
  } catch(Exc2 e) {
    System.out.println(4);
  } finally {
    System.out.println(5);
  }
}
```

Consider further the class declaration below.

```
class Exc2 extends Exception { }
class Exc1 extends Exc2 { }
```

Which of the following will be printed then calling `f()`?

A. 1
   2
   5
B. 1
   3
   5
C. 1
D. 1
   5
E. 1
   4
   5

5. Consider the code fragment below.

```
public static boolean foo(int x, int y) {
  Integer objX = x;
  Integer objY = y;
  return objX == objY;
}
```

Which of the following statement is true about the function above when called with `foo(n, n)`?

A. The function always return `false`.
B. The function always return `true`.
C. The function may return `true` or `false`.
D. If replaced with return `objX.equals(objY)`, the function always return `false`.
E. If replaced with return `objX.equals(objY)`, the function may return `true` or `false`.

Short Answer **[6 Marks]**

6. Consider the classes declarations below.

```
class A {}
class B {}
class C extends B {}
class D extends B {}
class E extends C {}
class F extends E {}

class P<T extends C> {}
```

Consider the variable declaration `P<? super F>  p`. Write down all the initialization of the form `p = new P<_____>()` that can be made without error where____is replaced with an actual class name from the list of classes above.

7. Consider the interface and abstract class below.

```
interface I {
  public void f();
  default void g() {}
}
abstract class A implement I {
  abstract public void h();
  abstract public void h(int x);
  public void j() {}
}
class B extends A { .. }
```

List all the methods and its signature (*including its access modifiers and return type*) that B needs to implement.

8. Consider the interface Comparable<T> with the usual method summary shown below.

`int compareTo(T o)` *Compares this object with the specified object for order.*

Consider further the class `Point` discussed in class, partially reproduced below for your convenience.

```
class Point implements Comparable<Point> {
  private double x;
  private double y;
  :
  public int compareTo(Point p) {
    return (int) (this.x - p.x);
  }
}
```

What will be the result of sorting the following array of `Point`? For simplicity, we write `<x1,y1>` to indicate a point with x-coordinate (i.e., `private double x`) of `x1` and y-coordinate (i.e., `private double y`) of `y1`.

`{ <3.1, 2.1>, <1.2, 2.2>, <2.2, 1.2>, <1.1, 2.1> }`

Long Answer **[9 marks]**

9. Consider the class `Rectangle` below.

```
class Rectangle {
  private int width;
  private int height;
  :
  public Rectangle(int height, int width) {
    this.width = width;
    this.height = height;
  }
}
```

Write a `toString` method for the class `Rectangle` above that prints the bounding box of the rectangle with minus (`-`), bar (`|`), and plus (`+`). For instance, `System.out.println(new Rectangle(3,5));` will produce the print out below. You are guaranteed that the minimum `width` and `height` for a rectangle is `2`. To insert a newline into a String, you use `“\n”`. For instance, `“a” + “\n”` will result in character `“a”` being printed followed by a newline.

```
+---+
|   |
+---+
```

10. Consider the generic class `Pair` below.

```
class Pair<T> {
  private T l;
  private T r;
  public Pair(T l, T r) {
    this.l = l;
    this.r = r;
  }
  public T getL() { return this.l; }
  public T getR() { return this.r; }
  public String toString() {
    return “<” + this.l + “,” + this.r + “>”;
  }
}
```

We want to create a generic class `Quad` which is an extension of `Pair` such that instead of holding a two elements of type `T`, it holds four elements of type `T`. We declare the class as follows: `class Quad<T> extends Pair<Pair<T>>`.

Without declaring any fields, implement the class `Quad`. You should refer to the following use case for the expected behaviour when run in `jshell`.

```
jshell> Quad<Integer> q = new Quad<>(1,2,3,4);
q ==> <<1,2>,<3,4>>

jshell> q.getL()
$17 ==> <1,2>

jshell> q.getR()
$18 ==> <3,4>
```

```
jshell> q.getLL()
$19 ==> 1

jshell> q.getLR()
$20 ==> 2

jshell> q.getRL()
$21 ==> 3

jshell> q.getRR()
$22 ==> 4
```

You should also implement the class ***minimally***. In other words, if a method is not needed in the class `Quad`, it should not be defined in the class `Quad`.