Unit 6 – ICS4U0 – ADTs

Sample Test – December 5, 2022

Names: Solutions

Total	%	Knowledge	Communication	Thinking	Application
(97)		(30)	(22)	(24)	(21)

Knowledge

1. Identify the object method type in each method call by writing in the appropriate first letter. /8
(C)onstructor, (M)utator, (A)ccessor, (F)acilitator

F	(a) int num = (int) Math.**random**()*5;
F	(b) FileOutputStream out = **openFileOutput**("txt.txt", Activity.MODE_PRIVATE);
C	(c) EditText et = **new EditText**();
A	(d) String words = et.**getText**();
M	(e) textArea.**setText**("Hello");
F	(f) if(question.**equals**(answer))
F	(g) if(word.**compareTo**(secondWord)>0)
A	(h) while(!s.**isFull**())

2. Circle the stack operations in this column.

dequeue
enqueue
insert
isEmpty
peek
pop
push

3. Circle the queue operations in this column. /4

dequeue
enqueue
insert
isEmpty
peek
pop
push

4. Note what is printed out beside each line of output. /8

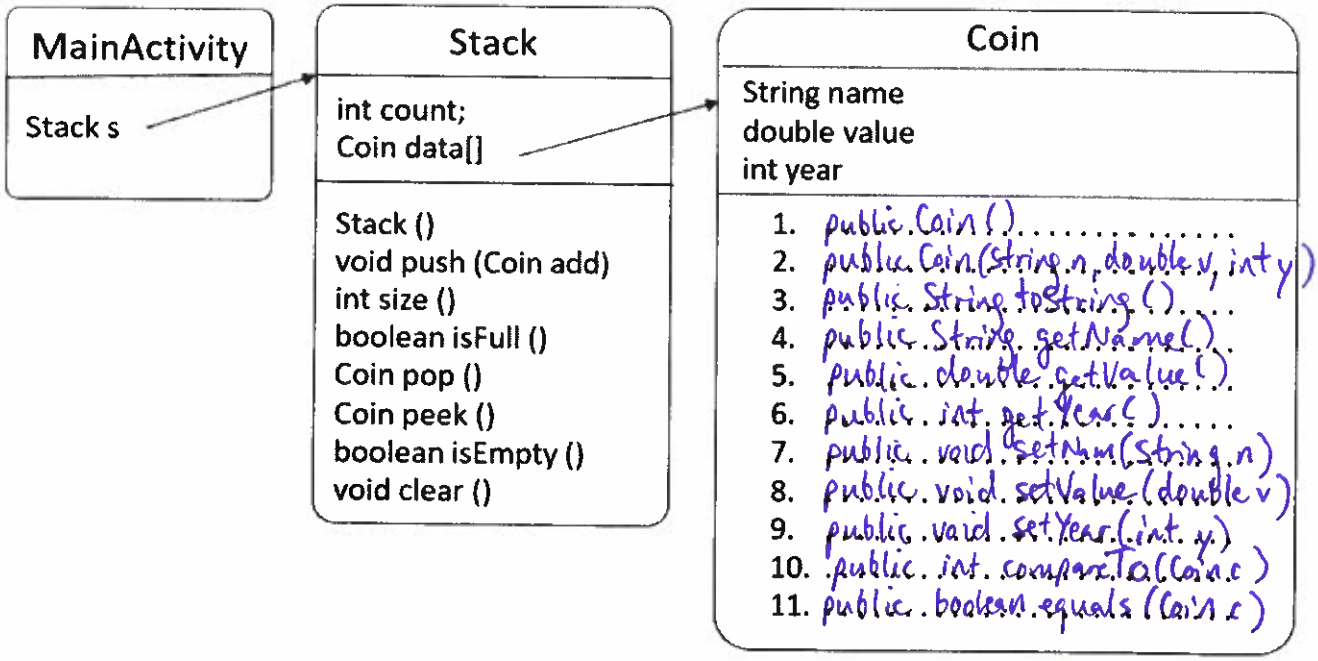
```
Queue q = new Queue ();
q.enqueue (9);
q.enqueue (8);
System.out.println (q.dequeue ());      9
System.out.println (q.size ());         1
q.enqueue (7);
q.enqueue (6);
System.out.println (q.dequeue ());      8
System.out.println (q.dequeue ());      7
System.out.println (q.isEmpty ());      false
```

```
Stack s = new Stack ();
s.push (9);
s.push (8);
System.out.println (s.pop ());          8
System.out.println (s.size ());         1
s.push (7);
s.push (6);
System.out.println (s.pop ());          6
System.out.println (s.pop ());          7
System.out.println (s.isEmpty ());      false
```

5. What does each acronym stand for? /4

Object	Abstract	Last	First
Oriented	Data	In	In
Programming	Type	First	First
		Out	Out

6. What 9 methods would be needed by the coin class? Fill them in the UML. /6



Communication

7. Identify the speeds of the following algorithms. /4

(a) pop O(1)
(b) push O(1)
(c) dequeue O(1)
(d) enqueue O(1)
(e) peek O(1)
(f) Binary search O(log n)
(g) Quicksort O(n log n)
(h) isFull O(1)

8. In the first column, fill in the terms that match the description. /8

instance	(a) Variables found inside an object's class.
private	(b) A keyword restricting variable access to within the class.
class	(c) A template for an object or type. Also contains a java program.
accessor	(d) An object method type that returns values of instance variables.
constructor	(e) An object method type that sets up dynamic memory.
mutator	(f) An object method type that changes the values of instance variables.
encapsulation	(g) Keeping an object's code self-contained and independent of other code. It relies only on itself.
instance variables	(h) Something that shouldn't be public.

9. Choose the best data structure for each description. /5

Stack Queue (a) A FIFO structure.
Stack Queue (b) A LIFO structure.
Stack Queue (c) A pile of books model this data structure well.
Stack Queue (d) The data structure that could model a waiting line.
Stack Queue (e) The data structure used by the browser's back button.

10. What is the trade-off associated with a Stack? /2

⊕ All stack operations (push, pop, peek, isEmpty etc) are O(1)
That is very fast.
⊖ A stack is limited to LIFO operations
→ We compromise on functionality to gain speed.

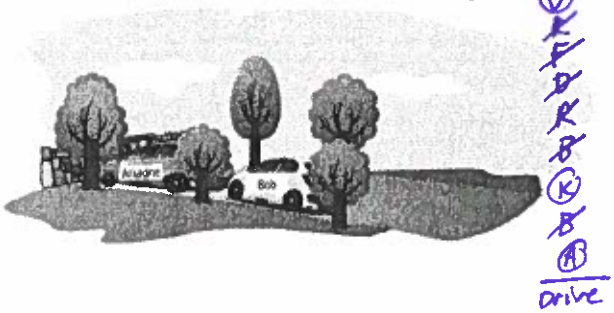
11. What are three characteristics of a class that supports information hiding? Explain each briefly. /3

① private instance variable - other coders using our class cannot access and change these directly. means they can't make a mistake and make the code unstable.
② accessors - allow other coders limited access to instance variables; only as keeps things stable.
③ mutators - only allows acceptable changes to instance variables.

Thinking

12. Neighbours share a long, narrow, driveway. Cars parked in the driveway leave by backing out. A schedule makes sure that nobody is blocked in when they need to leave. On each day, departing cars leave before any cars enter. Before Monday, no cars are in the driveway. The table shows how the driveway is shared. /2

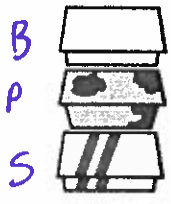
The driveway at the end of Monday is shown below:



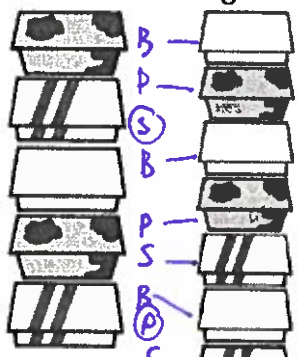
Day	# Leaving	# Entering	Owners in enter order
Mon	0	2	Ariadne, Bob
Tues	1	3	Kate, Ben, Roy
Wed	2	1	Daisy
Thurs	0	2	Finn, Rose
Fri	3	1	Vincent

Whose cars will be parked at the end of Friday? (circle)
(A) Bob, Vincent, Daisy (B) Vincent, Ariadne, Rose
(C) Ariadne, Kate, Vincent (D) Ariadne, Daisy, Vincent

13. Alan and Turing are working at a burger restaurant. Alan cooks burgers one at a time. After cooking a burger, he places it into one of three different boxes: one with stripes, one with a pattern and one plain box. If he has cooked three burgers, he would have a stack as shown in diagram A. If he cooked two more burgers, he would have a stack as shown in diagram B. /2



A



B

C

As Alan cooks a burger, he places that box on the top of the stack of not yet sold burgers and continues to cycle through the three different boxes (stripe, pattern, plain, stripe, pattern, plain, ...) to place the burger. Turing sells the burgers one at a time and always takes the uppermost box from the stack. Alan is cooking faster than Turing can sell the burgers. After some time, Turing has sold some burgers and Alan has cooked more burgers.

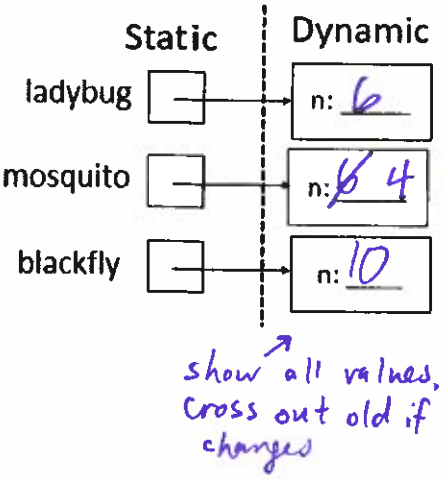
Suppose the stack of unsold burgers looks as shown in diagram C. What is the fewest number of burgers sold by Turing? (circle)
(a) 4
(b) 5
(c) 6
(d) 7

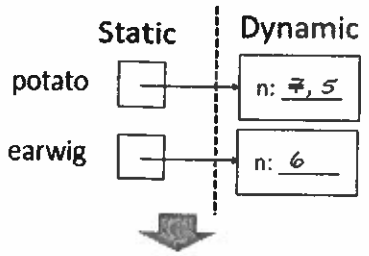
14. Trace this class: fill in both the output (on the lines) and the object diagram. /20

```
public class bug {
   private int n;
   public bug () {
      n = 6;
   }
   public bug (int r) {
      n = r;
   }
   public int buzz () {
      return n + 2;
   }
   public int fly () {
      return n;
   }
   public void setB(int r){
      n = r;
   }
   public void minusTwo () {
      n-=2;
   }
   public String toString () {
      return ""+ n;
   public boolean equals (bug b) {
      return b.buzz () == (n + 2);
   }
   public int compareTo (bug b) {
      if ((n + 2) == b.buzz ())
         return 0;
      else if ((n + 2) > b.buzz ())
         return 1;
      else
         return -1;
   }}
```

```
bug ladybug = new bug ();
System.out.println(ladybug.toString ());
bug mosquito = new bug (6);
System.out.println(mosquito.toString ());
System.out.println(ladybug.buzz ());
bug blackfly = new bug (10);
System.out.println(blackfly.toString ());
System.out.println(mosquito.equals (ladybug));
System.out.println(blackfly.compareTo (mosquito));
System.out.println(mosquito.compareTo (blackfly));
System.out.println(mosquito.fly());
mosquito.minusTwo();
System.out.println(mosquito.fly());
blackfly.buzz();
System.out.println(blackfly.fly());
```

6
6
8
10
true
1
-1
6
4
10





15. Using the above bug class and the above object diagram, write the code to produce the output shown.

Printed on the screen	***Code***

(declare the two objects)	bug potato = new bug (7);
	bug earwig = new bug ();
	S.o.p (potato);

7	S.o.p (potato);
8	S.o.p (earwig.buzz());
5	potato.minusTwo(); S.o.p (potato);
6	S.o.p (earwig);

Application

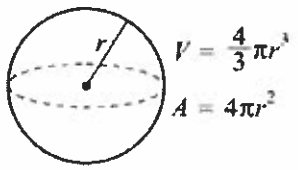
16. Fill in the Sphere class, then edit the Queue class to make a Queue of Spheres. /16

```
public class Sphere {
  private int radius;
  public Sphere() {
    radius = 5;
  }
  public Sphere(int r) {
    radius = r;
  }
  public void setRadius(int r) {
    radius = r;
  }
  public int getRadius() {
    return radius;
  }
  public String toString() {
    return "The sphere has radius " +
           radius + " units.";
  }
  public double surfaceArea() {
    return 4*Math.PI*radius*radius;
  }
  public double volume() {
    return 4 * Math.PI * radius
      * radius * radius / 3;
  }
  public boolean equals(Sphere s) {
  //return true if equal, false otherwise
    if(radius == s.getRadius())
      return true;
    else
      return false;
  }
  public int compareTo(Sphere s) {
  //returns 1 if radius is biggest
    if(radius>s.getRadius())
      return 1;
    //returns 0 if equal
    else if (radius==s.getRadius())
      return 0;
    //return -1 otherwise
    else
      return -1;
  }
}
```

5 can be something else

```
public class Queue {
  private Object data[] = new Object [50];
  int count;
  int head;

  public Queue () {
    count = 0;
    head = 0;
  }

  public void enqueue (Object value) {
    int tail = (head + count) % data.length;
    data [tail] = value;
    count++;
  }

  public Object dequeue () {
    Object temp = data [head];
    count--;
    head = (head + 1) % data.length;
    return temp;
  }

  public Object peek () {
    return data [head];
  }

  public int size () {
    return count;
  }

  public boolean isEmpty () {
    return (count == 0);
  }
}
```

Sphere Sphere Sphere Sphere Sphere



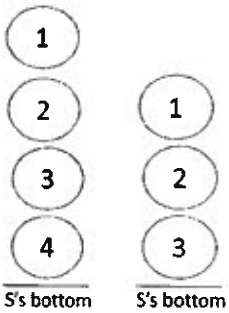
Call your new Queue in the MainActivity class.

```
public void makeQueue(View view) {
    Queue q = new Queue ();
    q.enqueue(new Sphere( 10 ));
    Sphere m = new Sphere( 7 );
    q.enqueue( m );
    q.dequeue();
}
```

whatever whatever

17. Remove and print out the bottom element of the Stack named 's'. After you are done, the rest of the Stack should be in its original order. The picture shows **an example** of this process.

/5



Prints: 4

```
Stack s2 = new Stack();
while (!s.isEmpty())
    s2.push(s.pop());

System.out.println(s2.pop());

while (!s2.isEmpty())
    s.push(s2.pop());
```

18. (BONUS: This is a question from pre COVID times) Using only stack and queue functions, determine if a Queue is a palindrome. Additional Stacks and Queues are permitted. The example shown is a palindrome.

```
Queue q = new Queue ();
q.enqueue (1);
q.enqueue (2);
q.enqueue (3);
q.enqueue (4);
q.enqueue (3);
q.enqueue (2);
q.enqueue (1);
```

```
Stack s = new Stack();
Queue q2 = new Queue();

while (!q.isEmpty()) {
    s.push(q.peek());
    q2.enqueue(q.dequeue());
}

boolean good = true;
while (!q2.isEmpty()){
    if (s.pop() != q2.peek())
        good = false;
    q.enqueue(q2.dequeue());
}

if (good)
    s.o.p ("Palindrome");
else
    s.o.p ("Not Palindrome");
```