CMP-338 Name:______________________________

Midterm Spring 2015 Solutions

Version 1

1. (2 Points) In order to declare a named constant, the declaration must use which Java keyword?
   - **a. final**
   - b. int
   - c. static
   - d. void

2. (2 Points) Suppose c1 and c2 are objects of the class Circle. A Circle has a single data member, its radius. The Circle class has a default constructor (implemented correctly), but no other methods have been defined in the implementation of the Circle class. What will happen when we try to execute this code?

   Circle c1 = new Circle(12.0);
   Circle c2 = new Circle(12.0);
   boolean same = (c1.equals(c2));

   - a. The code will not compile because equals( ) has not been implemented in Circle.
   - b. The value of same will be true.
   - **c. The value of same will be false.**

3. (2 Points) The Java expression 9 / 5 + 9 % 5 equals ______.
   - a. 0
   - b. 1
   - c. 3
   - **d. 5**
   - e. 6

4. (2 Points) Which of the following is a base case for a recursive binary search algorithm? (first is the index of the first item in the array, last is the index of the last item in the array, and mid is the midpoint of the array).
   - a. last > first
   - **b. first > last**
   - c. 0 <= first
   - d. last <= SIZE-1

5. (2 Points) In the following list:
   John, Kate, Fred, Mark, Jon, Adam, Drew
   which element is the head of the list?
   - **a. John**
   - b. Mark
   - c. Drew
   - d. Adam

6. (2 Points) Which of the following statements deletes the node that curr references?
   - a. prev.setNext(curr);
   - b. curr.setNext(prev);
   - c. curr.setNext(curr.getNext());
   - **d. prev.setNext(curr.getNext());**

7. (20 Points) Given the following StackInterface:

```
public interface StackInterface {
   public void push(Object obj);
   public Object pop();
}
```

The correct array-based implementation is:

```
public class ArrayStack implements StackInterface {

   private final int MAX_STACK = 1000;
   private final int INVALID_STACK_POINTER = -1;

   private Object[] arrayStack = new Object[MAX_STACK];
   private int stackPointer = INVALID_STACK_POINTER;

   @Override
   public void push(Object obj) {
         if (++stackPointer < MAX_STACK) {
               arrayStack[stackPointer] = obj;
         }
   }

   @Override
   public Object pop() {
         Object obj = null;
         if (stackPointer != INVALID_STACK_POINTER) {
               obj = arrayStack[stackPointer--];
         }
         return obj;
   }
}
```

8. (30 Points) The correct ReferenceQueue implementation is:

```
public class ReferenceQueue implements QueueInterface {

    private Node front = null, back = null;

    @Override
    public void add(Object obj) {
            Node newNode = new Node(obj);
            if (back == null) {
                    front = newNode;
                    back = front;
            } else {
                    back.setNext(newNode);
                    back = newNode;
            }
    }

    @Override
    public Object remove() {
            Object obj = null;
            if (front != null) {
                    obj = front.getObject();
                    front = front.getNext();
            }
            if (front == null) {
                    back = null;
            }
            return obj;
    }
}
```

9. (25 Points) The correct Java method:

```
    public static void sort(int[] numbers) {
```

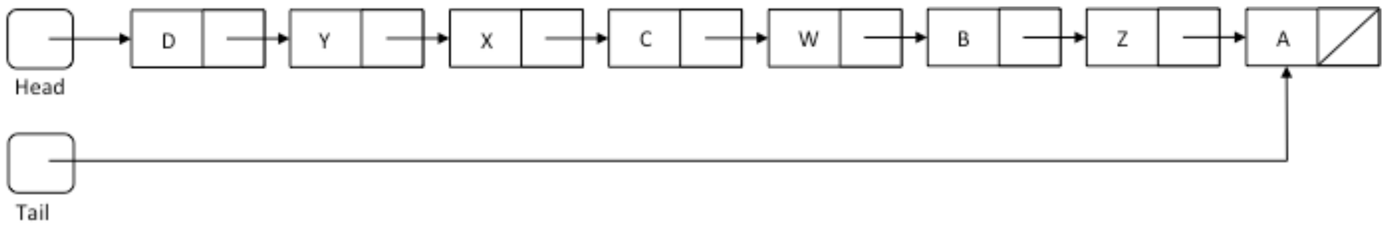
which sorts the `numbers` array using Bubble Sort is:

```
    public static void sort(int[] numbers) {
       int lastSwap;
       int lastCell = numbers.length -1;
       int temp;

       do {
          lastSwap = 0;

          for ( int i = 0 ; i < lastCell ; i++ ) {
             if (numbers[i] > numbers[i+1]) {
                temp = numbers[i];
                numbers[i] = numbers[i+1];
                numbers[i+1] = temp;
                lastSwap = i;
             }
          }
          lastCell = lastSwap;
       } while(lastCell > 0);
    }
```

10. (20 Points) The list after `doStuff()` has finished executing:



CMP-338 Name:______________________________

Midterm Spring 2015 Solutions
Version 2

1. (2 Points) If x is a variable of type int, what is the largest possible value of the expression (x % 5)?

    a. 1<br>
    **b. 4**<br>
    c. 5<br>
    d. $2^{31} - 1$

2. (2 Points) Which type of loop is guaranteed to execute its body at least once?

    **a. do-while**<br>
    b. for<br>
    c. switch<br>
    d. while

3. (2 Points) If we wanted to write an if-statement that executes whenever the real number x is between 10.0 and 20.0, how should the test condition be written?

    a) 10.0 < x || x > 20.0<br>
    b) 10.0 < x && x > 20.0<br>
    **c) 10.0 < x && x < 20.0**<br>
    d) 10.0 < x || x < 20.0

4. (2 Points) A recursive solution that finds the factorial of n generates _____ recursive calls.

    a. n-1<br>
    **b. n**<br>
    c. n+1<br>
    d. n*2

5. (2 Points) In the following list:
John, Kate, Fred, Mark, Jon, Adam, Drew
which element is the tail of the list?

    a. John<br>
    b. Mark<br>
    **c. Drew**<br>
    d. Adam

6. (2 Points) If a linked list is empty, the statement head.getNext() will throw a(n) _______.

    a. IllegalAccessException<br>
    b. ArithmeticException<br>
    c. IndexOutOfBoundsException<br>
    **d. NullPointerException**

7. (20 Points) Given the following StackInterface:

```java
public interface StackInterface {
    public void push(Object obj);
    public Object pop();
}
```

The correct referenced-based implementation is:

```java
public class Node {
    private Object object;
    private Node next;

    public Node(Object object) {
        this.object = object;
        this.next = null;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }

    public Object getObject() {
        return object;
    }
}
```

```java
public class ReferenceStack implements StackInterface {

    private Node stackPointer = null;

    @Override
    public void push(Object obj) {
        Node newNode = new Node(obj);
        if (stackPointer == null) {
            stackPointer = newNode;
        } else {
            newNode.setNext(stackPointer);
            stackPointer = newNode;
        }
    }

    @Override
    public Object pop() {
        Object obj = null;
        if (stackPointer != null) {
            obj = stackPointer.getObject();
            stackPointer = stackPointer.getNext();
        }
        return obj;
    }
}
```

Midterm Spring 2015 Solutions
Version 2

8. (30 Points) The correct ArrayQueue implementation is:

```
import java.util.Vector;

public class ArrayQueue implements QueueInterface {

    private Vector<Object> arrayQueue = new Vector<Object>();

    @Override
    public void add(Object obj) {
            arrayQueue.addElement(obj);
    }

    @Override
    public Object remove() {
            Object obj = null;
            if (arrayQueue.size() > 0) {
                    obj = arrayQueue.elementAt(0);
                    arrayQueue.remove(0);
            }
            return obj;
    }
}
```

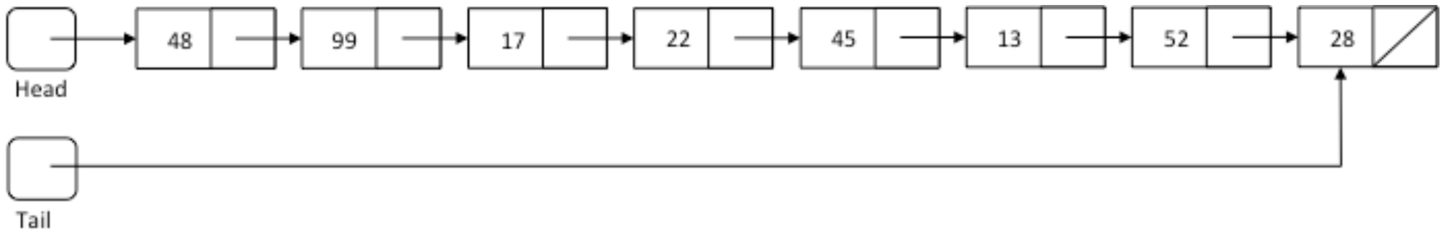
9. (25 Points) The correct Java method:

```
public static void sort(int[] numbers) {
```

which sorts the `numbers` array using Selection Sort is:

```
public static void sort(int[] numbers) {
   for ( int i = 0 ; i < numbers.length ; i++) {
      int curMin = numbers[i];
      int curMinIndex = i;
      for ( int j = i+1 ; j < numbers.length ; j++) {
         if (numbers[j] < curMin) {
            curMin = numbers[j];
            curMinIndex = j;
         }
      }
      numbers[curMinIndex] = numbers[i];
      numbers[i] = curMin;
   }
}
```

10. (20 Points) The list after `doStuff()` has finished executing:



CMP-338 Name:______________________________

Midterm Spring 2015 Solutions
Version 3

1. (2 Points) Which of the following loop headers will arrange for the loop body to execute exactly 10 times?
   a. for (int i = 1; i < 10; ++i)
   b. for (int i = 0; i <= 10; ++i)
   **c. for (int i = –5; i < 5; ++i)**
   d. for (int i = 2; i < 20; ++i)

2. (2 Points) If s1 is of type String, what does s1.compareTo(s1) return?
   **a. zero**
   b. true
   c. false
   d. Cannot be determined without knowing the value of s1.

3. (2 Points) Consider the following code that appears in a test class.
   A a = new A();
   int c = a.b;
   In order for this code to work, which statement must be true?
   a. a must be declared public inside class A
   **b. b must be declared public inside class A**
   c. c must be declared public inside class A
   d. Method b( ) must return int

4. (2 Points) If the value being searched for by a recursive binary search algorithm is in the array, which of the following is true?
   a. the algorithm cannot return a nonpositive number
   b. the algorithm cannot return a nonnegative number
   c. the algorithm cannot return a zero
   **d. the algorithm cannot return a negative number**

5. (2 Points) In the following list:
   John, Kate, Fred, Mark, Jon, Adam, Drew
   which element does not have a predecessor?
   **a. John**
   b. Mark
   c. Drew
   d. Kate

6. (2 Points) Which of the following will be true when the reference variable curr references the last node in a linear linked list?
   a. curr == null
   b. head == null
   **c. curr.getNext() == null**
   d. head.getNext() == null

7. (20 Points) Given the following QueueInterface:

```
public interface QueueInterface {
   public void add(Object obj);
   public Object remove();
}
```

The correct array-based implementation is:

```
import java.util.Vector;

public class ArrayQueue implements QueueInterface {

   private Vector<Object> arrayQueue = new Vector<Object>();

   @Override
   public void add(Object obj) {
      arrayQueue.addElement(obj);
   }

   @Override
   public Object remove() {
      Object obj = null;
      if (arrayQueue.size() ≥ 0) {
         obj = arrayQueue.elementAt(0);
         arrayQueue.remove(0);
      }
      return obj;
   }
}
```

8. (30 Points) The correct ReferenceStack implementation is:

```
public class ReferenceStack implements StackInterface {

    private Node stackPointer = null;

    @Override
    public void push(Object obj) {
            Node newNode = new Node(obj);
            if (stackPointer == null) {
                    stackPointer = newNode;
            } else {
                    newNode.setNext(stackPointer);
                    stackPointer = newNode;
            }
    }

    @Override
    public Object pop() {
            Object obj = null;
            if (stackPointer != null) {
                    obj = stackPointer.getObject();
                    stackPointer = stackPointer.getNext();
            }
            return obj;
    }
}
```

9. (25 Points) The correct Java method:

```
public static void sort(int[] numbers) {
```

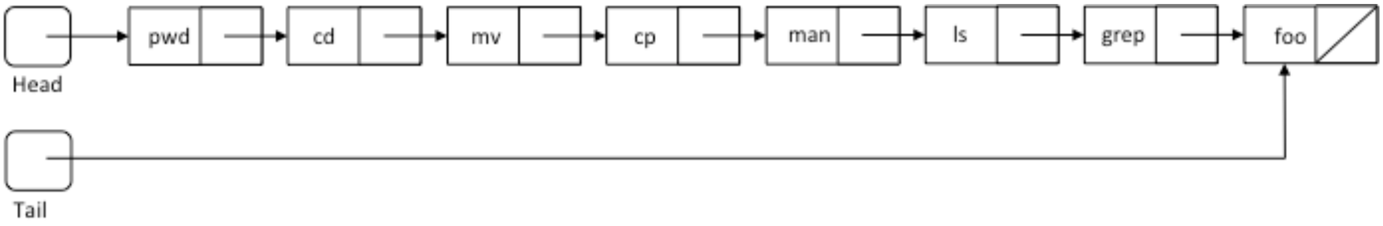
which sorts the `numbers` array using Bubble Sort is:

```
public static void sort(int[] numbers) {
   int lastSwap;
   int lastCell = numbers.length -1;
   int temp;

   do {
      lastSwap = 0;

      for ( int i = 0 ; i < lastCell ; i++ ) {
         if (numbers[i] > numbers[i+1]) {
            temp = numbers[i];
            numbers[i] = numbers[i+1];
            numbers[i+1] = temp;
            lastSwap = i;
         }
      }
      lastCell = lastSwap;
   } while(lastCell > 0);
}
```

10. (20 Points) The list after `doStuff()` has finished executing:



CMP-338 Name:_______________________________

Midterm Spring 2015 Solutions
Version 4

1. (2 Points) What type of Java statement allows you to use classes contained in other packages?
   a. an access statement
   b. a class statement
   **c. an import statement**
   d. a package statement

2. (2 Points) How many constructors can a class have?
   a. Exactly one
   b. At least one but no more than three
   c. Exactly the same as the number of data members
   **d. There is no restriction on the number of constructors**

3. (2 Points) Which of these expressions is illegal in Java?
   **a. x++ 5**
   b. x =+ 5
   c. x += 5
   d. x == 5

4. (2 Points) The midpoint of a sorted array can be found by ______, where first is the index of the first item in the array and last is the index of the last item in the array.
   a. first / 2 + last / 2
   b. first / 2 – last / 2
   **c. (first + last) / 2**
   d. (first – last) / 2

5. (2 Points) In the ADT list, when an item is deleted from position i of the list, ______.
   a. the position of all items is decreased by 1
   b. the position of each item that was at a position smaller than i is decreased by 1
   **c. the position of each item that was at a position greater than i is decreased by 1**
   d. the position of each item that was at a position smaller than i is increased by 1 while the position of each item that was at a position greater than i is decreased by 1

6. (2 Points) The last node of a linear linked list ______.
   a. has the value null
   **b. has a next reference whose value is null**
   c. has a next reference which references the first node of the list
   d. cannot store any data

7. (20 Points) Given the following QueueInterface:

```
public interface QueueInterface {
    public void add(Object obj);
    public Object remove();
}
```

The correct referenced-based implementation is:

```
public class Node {
    private Object object;
    private Node next;

    public Node(Object object) {
        this.object = object;
        this.next = null;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }

    public Object getObject() {
        return object;
    }
}
```

```
public class ReferenceQueue implements QueueInterface {

    private Node front = null, back = null;

    @Override
    public void add(Object obj) {
        Node newNode = new Node(obj);
        if (back == null) {
            front = newNode;
            back = front;
        } else {
            back.setNext(newNode);
            back = newNode;
        }
    }

    @Override
    public Object remove() {
        Object obj = null;
        if (front != null) {
            obj = front.getObject();
            front = front.getNext();
        }
        if (front == null) {
            back = null;
        }
        return obj;
    }
}
```

Midterm Spring 2015 Solutions
Version 4

8. (30 Points) The correct ArrayStack Implementation is:

```
public class ArrayStack implements StackInterface {

    private final int MAX_STACK = 1000;
    private final int INVALID_STACK_POINTER = -1;

    private Object[] arrayStack = new Object[MAX_STACK];
    private int stackPointer = INVALID_STACK_POINTER;

    @Override
    public void push(Object obj) {
            if (++stackPointer < MAX_STACK) {
                    arrayStack[stackPointer] = obj;
            }
    }

    @Override
    public Object pop() {
            Object obj = null;
            if (stackPointer != INVALID_STACK_POINTER) {
                    obj = arrayStack[stackPointer--];
            }
            return obj;
    }
}
```

9. (25 Points) The correct Java method:

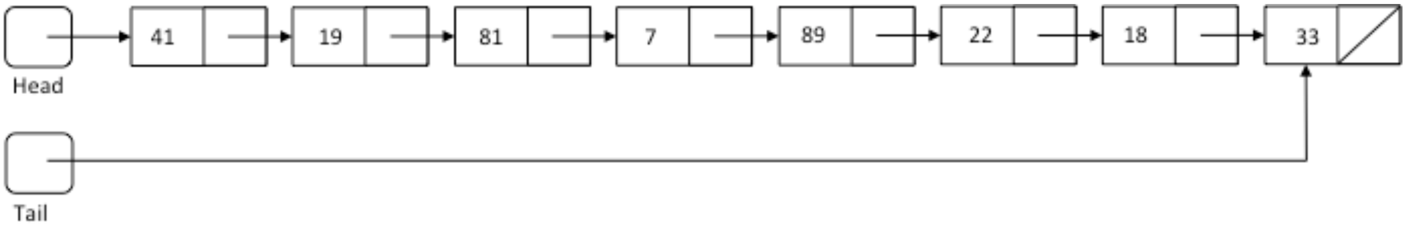
```
public static void sort(int[] numbers) {
```

which sorts the `numbers` array using Insertion Sort is:

```
public static void sort(int[] numbers) {
   int j, temp;
   for (int i = 1; i < numbers.length; i++) {
      temp = numbers[i];
      j = i;

      while ((j > 0) && (numbers[j - 1] > temp)) {
         numbers[j] = numbers[j - 1];
         j -= 1;
      }
      numbers[j] = temp;
   }
}
```

10. (20 Points) The list after `doStuff()` has finished executing:


41
19
81
7
89
22
18
33
Head
Tail