CS 307 – Midterm 1 – Spring 2002

Name____________________________________

Last 4 digits of SSN / Student ID ______________

Class Unique ID ___________________________

Instructions:
1. There are 4 questions on this test.
2. You will have 2 hours to complete the test.
3. You may not use a calculator.
4. When code is required, write Java code.
5. Please make your answers legible.
6. The style guide is not in effect except as noted.

1. (2 points each, 30 points total) Java Mechanics. For all parts, what is the output of the code fragment? Write your answer on the line provided. Consider each piece of code in isolation. If an error would occur answer "syntax error" or "runtime error" depending on what type of error it would be.

A. _________________________________________________

```java
int limit = 20;
int[] intList = new int[limit / 3];

for (int i = 0; i < intList.length; i++)
    if (i % 3 == 0)
        intList[i] = i;
    else
        intList[i] = -1;

for (int i = 0; i < intList.length; i++)
    System.out.println( intList[i] );
```
B.

```java
int x = 34;
int y = 45;
int z = 123;

double answer;

answer = ( x / y ) * (double)( y / z );

System.out.println( answer );
```

C.

```java
int i = 0;
int j = 4 / i;
System.out.println( "j = " + j );
```

D.

```java
double a = 0.0;
double b = 2.5;
int x = 13;
int y = 28;

a = b + y % x;

System.out.println( a );
```

E.

```java
// The BigInteger class is part of the java.lang
// package. It is a class that can be used to
// represent arbitrary precision integers,
// integer much larger than int or long can
// represent. Passing a String to the BigInteger
// constructor creates a new BigInteger object
// representing the argument

BigInteger b1 = new BigInteger("1212");
BigInteger b2 = new BigInteger("1212");

System.out.println( b1 == b2 );
```
F. _________________________________________________

// two BigInteger Objects are "equal" if the
// integer values they are representing are equal.

BigInteger b1 = new BigInteger("1212");
BigInteger b2 = new BigInteger("1212");

System.out.println( b1.equals(b2) );

G. _________________________________________________

Assume the following two methods appear in the same class, Foo.

```java
public int chalk( int m, int p )
{ int t = m;
  m = m + 2 * p;
  return t + m + p;
}

public void marker( int x, int y )
{ x = x + 2;
  y++;

  System.out.println( chalk(x, y) + " " + x + " " + y );
}
```

What is the output of the following method call?

someFooObject.marker(2, 4);
For parts H and I consider these three classes:

```java
class Mammal {
    public String birthType() {
        return "Live Birth.";
    }

    // other methods and instance vars not shown
}

class Platypus extends Mammal {
    public String birthType() {
        return "Lays eggs.";
    }

    // other methods and instance vars not shown
}

class Whale extends Mammal {
    // no method named birthType declared in class Whale

    // other methods and instance vars not shown
}
```

H. _________________________________________________

What is output by the following code?

```java
Platypus george = new Platypus();
System.out.println( george.birthType() );
```

I. _________________________________________________

What is output by the following code?

```java
Whale skippy = new Whale();
System.out.println( skippy.birthType() );
```
For part J consider the following class:

```java
public class Strange
{
    public String name;
    public int x;
}
```

These two methods appear together in a class other than Lousy:

```java
public void dolphin( Strange one, Strange two )
{
    one.name = two.name;
    two.name = "Olivia";
    one.x = 28;
}

public void stingray()
{
    Strange a = new Strange();
    a.name = "Kelly";
    a.x = 12;
    Strange b = new Strange();
    b.name = "David";
    b.x = 20;
    dolphin( a, b );
    System.out.println( a.name + " " + a.x );
}
```

J. __________________________________________________________________________

What is output when method stingray is called?

K. __________________________________________________________________________

```java
double a = 3 * 5 + 2.5 - 15 / 2;
System.out.println( a );
```

L. __________________________________________________________________________

```java
int[] list = { 2, 3, 5, 7, 11, 13, 17 };
int x = list[ list.length - 1 ] * list[ 3 / 5 ] + list[ list[0] ];
System.out.println( x );
```
M.  

// new String("Dog") creates a new String object that contains the  
// string "Dog"

String[] slist = new String[5];

for(int i = 0; i < slist.length; i++)
    slist[i] = new String("Cat");
slist[2] = "Mud";

int total = 0;
for(int j = 0; j < slist.length; j++)
    total += slist[j].length();  
// the String method length() returns the number of characters in  
// the String object

System.out.println( total );

N.  

int[] list;
System.out.println( list.length );

O.  

int limit = 6;
int total = 0;
for(int i = 0; i < limit; i++)
{   total += i;
    limit--;
}
System.out.println( total );
This page intentionally blank. You may use it for scratch paper.
2. (25 points) Imagine a grocery store has unknowingly hired Timmy the evil Cracker as a stock boy. Timmy changes the store's computerized inventory each night to impress his friends on AOL. Each morning, the grocery store staff must rebuild the inventory. Complete the following method, `rebuildInventory`, which undoes Timmy's changes. The parameter `inv` contains all the StockItem objects that make up the store's inventory (the `inv` array is full). Each type of item the store carries should be represented by only one StockItem object in `inv`. However, in some cases, Timmy has put multiple StockItem objects into `inv` for each item. The method should merge together any StockItem objects in `inv` that represent the same item, and return a new array of StockItems in which each item only appears once, with the proper quantity.

For example, if `inv` contains the following data before method `rebuildInventory` is called:

```
index: |   0   |   1   |   2   |   3   |   4   |
Code:  | 12837 | 00342 | 12837 | 01343 | 01343 |
Price: | $3.29 | $9.95 | $3.29 | $8.34 | $8.34 |
Quantity: | 34 | 160 | 98 | 8 | 22 |
```

Then `rebuildInventory` should return an array like this:

```
index: |   0   |   1   |   2   |
Code:  | 12837 | 00342 | 01343 |
Price: | $3.29 | $2.95 | $8.34 |
Quantity: | 132 | 160 | 30 |
```

The order of the StockItems in the returned array could vary, but notice how each StockItem appears only once (based on Codes) and the quantity is correct.

Here is the public interface for the StockItem class:

```
public class StockItem
{
    // pre: code != null, price > 0, quantity > 0
    // post getCode() = code, getPrice() = price,
    // getQuantity = quantity
    public StockItem(String code, double price, int quantity)

    // pre: none
    // post: return code
    public String getCode()

    // pre: none
    // post: return price
    public double getPrice()

    // pre: none
    // post: return quantity
    public int getQuantity()
}
```

Do not assume any other methods are in the StockItem class.
Explain your algorithm in English below. This is worth 5 points.

Complete the following method as described on the previous page. This method is not part of the StockItem class

/* pre: inv != null, inv[n] != null for n = 0 to (inv.length - 1)
   You may also assume any StockItems whose codes are equal will
   have the same price
post: return an array of StockItems such that no item with
   the same product code appears more than once in the array and
   the quantity of each item in the returned array is equal to
   the total quantity of items with the same code in inv. The
   returned array shall not have any null pointers in it. */

public StockItem[] rebuildInventory( StockItem[] inv )
{

// more space available on next page
Intentionally blank. Complete `rebuildInventory` here if more space is needed.
3. (20 points) Implement a class to model cars in a traffic simulation. Each car has a speed which will be an integer and an ID number, also an integer. Speed is not in miles per hour. The units are different, but unstated. Complete a default constructor and a constructor where initial speed and ID number may be specified. For the following methods be sure you use the names specified. 

getSpeed and getId should both return ints. The accelerate method increases a car's speed by 1 unit length per unit time. So if getSpeed() = 3 then after calling accelerate once getSpeed() = 4. Car's cannot accelerate to a speed greater than the maximum speed. The decelerate method takes an integer parameter and reduces speed by that amount, not to go below 0. A car's maximum speed is 5, so 0 <= car's speed <= 5 at all times. Also write a toString method and an equals method. Two cars are considered equal if their speeds are the same. Ensure all instance variables are private and use the my iMy prefix.
4. (25 points) Write a method to simulate traffic moving along a stretch of highway named step. The highway is represented by an array of Car objects from question number 3. You may assume your Car class works for this question. This method is not part of the Car class. Any element of the highway (array) that is not currently occupied by a car is equal to null. Movement of cars occurs from the 0 index towards the length – 1 index of the array. Here is an example:

Car 1 in element 0 has a speed of 1, car 2 in element 3 has a speed of 3, and car 3 in element 5 has a speed of 3.

Your method will simulate 1 time unit of movement. All cars first check to see if they can accelerate or if they must maintain speed or decelerate. To do this cars look forward, towards the end of the array. They do not examine other cars' speeds. Acceleration and deceleration occur before movement. Cars use the following rules to govern speed:

A. If possible accelerate by one unit of speed. To do this all spaces must be clear from the current space forward to the destination space. For example car 1 is at a speed of 1. If it accelerates its speed will be 2. When the time for movement comes, it is safe to move forward two spaces to element 2. Note, all the spaces between the car and its projected location must be clear for the acceleration to take place. If a car is already at maximum speed it should not try to accelerate any more. Also note, if element 3 were null car 1 would still only accelerate to a speed of 2, because cars will only accelerate by one unit of speed at a time.

B. If the car cannot accelerate due to cars in its path then it must either maintain speed or decelerate. Cars in the path are based on current positions, not projected positions. This is to simulate cautious drivers. For example if there were a car in element 1 then Car 1 in element 0 would have to decelerate to a speed of 0, regardless of the speed of the car in element 1. Note this rule affects car 2 in element 3. Its speed is 3, which would place it in element 6. However, there is a car in element 5, so car 2 must decelerate to a speed of 1. This will put it in a safe spot, element 4 when the time to move comes.

C. If a car's speed would take it off the highway and the array is clear then the car is assumed to exit the portion of the highway the array represents and it is removed from the array.

After adjusting all cars' speeds the cars are moved along the array. Each unit of speed moves a car one element forward in the array. Given the previous array, after method step the array would look like this:

index: 0 | 1 | 2 | 3 | 4 | 5 | 6
value: null null null | null | null null
Cars - - -
Speed: 2 1
ID: 1 2
The key thing to remember is a car's speed is adjusted based on the current position of the other cars, not the anticipated position of the other cars.

Complete the following method as described on the previous page. Remember it is not a part of the Car class

```java
public void step( Car[] highway )
{"