ECE 39595J Fall 2020, Exam 3

The answer sheet is a separate sheet and can be edited, and therefore annotated with your answers.

Not counting the answer sheet, this exam has 39 questions and 13 pages.

You may begin the exam whenever it becomes available. I will give a 10 minute warning at the end of that the exam answer sheet needs to have already been uploaded to Brightspace.

If you are not in zoom with video turned on you may recieve a 0 on the exam. I will be recording the exam. Check the Zoom chat box periodically for corrections.

Programs are be given without import statements for brevity. Assume all needed imports are present. You may use newlines in your answer, or not, without affecting your score.

Each question is worth 2.5 points.

This program has questions 1 - 5. If something is printed by a line that is a question (has a Qx comment, where "x" is a number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist when answering other questions. If the statement prints nothing but is correct, answer "Ok". If a value is unitialized, answer "uninit".

```
public class IntArray implements Cloneable {

   public IntArray(int len) {
      ary = new int[len];
      reset( );
   }

   public int getElement(int i) {
      return ary[i];
   }

   public void setElement(int i, int val) {
      ary[i] = val;
   }

   public void reset( ) {
      for (int i = 0; i < ary.length; i++) {
         ary[i] = i;
      }
   }

   public Object clone( ) throws CloneNotSupportedException {
      return super.clone( );
   }
   private int[ ] ary = null;
}
public class Test {

   public static void main(String[ ] args) throws CloneNotSupportedException {
      IntArray a = new IntArray(3);
      IntArray b = a;
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q1
      a.setElement(0,2);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q2
      a.reset( );
      b = a.clone( ); // Q3
      b = (IntArray) a.clone( );
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q4
      a.setElement(0,2);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q5
   }
}
```

This program has questions 6 - 7. If something is printed by a line that is a question (has a Qx comment, where "x" is a number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist when answering other questions. If the statement prints nothing but is correct, answer "Ok". If a value is unitialized, answer "uninit".

```
public class IntArray {

   public IntArray(int len) {
      ary = new int[len];
      reset(this);
   }

   public IntArray(IntArray ia) {
      ary = ia.ary;
   }

   public int getElement(int i) {
      return ary[i];
   }

   public void setElement(int i, int val) {
      ary[i] = val;
   }

   public static void reset(IntArray ia) {
      for (int i = 0; i < ia.ary.length; i++) {
         ia.ary[i] = i;
      }
   }
   private int[ ] ary = null;
}
public class Test {

   public static void main(String[ ] args) throws CloneNotSupportedException {
      IntArray a = new IntArray(3);
      IntArray b = a;
      b = new IntArray(a);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q6
      a.setElement(0,2);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q7
   }
}
```

This program has questions 8 - 9. If something is printed by a line that is a question (has a Qx comment, where "x" is a number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist when answering other questions. If the statement prints nothing but is correct, answer "Ok". If a value is unitialized, answer "uninit".

```
public class IntArray implements Cloneable {

   public IntArray(int len) {
      ary = new int[len];
      reset(this);
   }

   public int getElement(int i) {
      return ary[i];
   }

   public void setElement(int i, int val) {
      ary[i] = val;
   }

   public static void reset(IntArray ia) {
      for (int i = 0; i < ia.ary.length; i++) {
         ia.ary[i] = i;
      }
   }

   public Object clone( ) throws CloneNotSupportedException {
      IntArray temp = (IntArray) super.clone( );
      temp.ary = new int[ary.length];
      for (int i = 0; i < ary.length; i++) {
         temp.ary[i] = ary[i];
      }
      return temp;
   }
   private int[ ] ary = null;
}
public class Test {

   public static void main(String[ ] args) throws CloneNotSupportedException {
      IntArray a = new IntArray(3);
      IntArray b = a;
      b = (IntArray) a.clone( );
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q8
      a.setElement(0,2);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q9
   }
}
```

This program has questions 10 - 12. If something is printed by a line that is a question (has a Qx comment, where "x" is a number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist when answering other questions. If the statement prints nothing but is correct, answer "Ok". If a value is unitialized, answer "uninit".

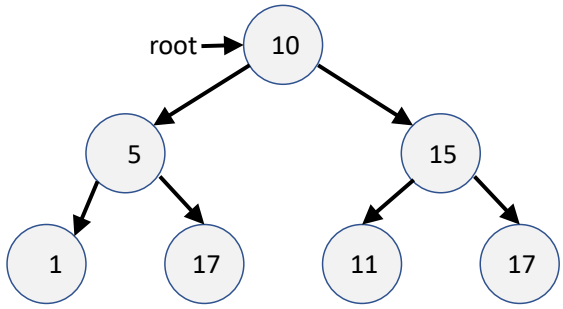
```
public class IntArray {

   public IntArray(int len) {
      ary = new int[len];
      reset(this);
   }

   public IntArray(IntArray ia) {
      ary = new int[ia.ary.length];
      for (int i = 0; i < ary.length; i++) {
         ary[i] = ia.ary[i];
      }
   }

   public int getElement(int i) {
      return ary[i];
   }

   public void setElement(int i, int val) {
      ary[i] = val;
   }

   public static void reset(IntArray ia) {
      for (int i = 0; i < ia.ary.length; i++) {
         ia.ary[i] = i;
      }
   }
   private int[ ] ary = null;
}
public class Test {

   public static void main(String[ ] args) throws CloneNotSupportedException {
      IntArray a = new IntArray(3);
      IntArray b = a;
      b = a.clone( ); // Q10
      b = new IntArray(a);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q11
      a.setElement(0,2);
      System.out.println(""+a.getElement(0)+", "+b.getElement(0)); // Q12
   }
}
```

This page has questions 13 – 14, below.

```
public class Node {

   private int data = -1;
   Node left = null;
   Node right = null;

   public Node(int val) {
      data = val;
   }

   public Node insertNode(int n) {
      if (n == data) return this;
      if (n < data) {
         if (left != null)
            return left.insertNode(n);
         else {
            left = new Node(n);
            return left;
         }
      }
      if (n > data) {
         if (right != null)
            return right.insertNode(n);
         else {
            right = new Node(n);
            return left;
         }
      }
      assert false : "node "+n+" not inserted";
      return null;
   }

   public String toString( ) {
      String str = "";
      if (left != null)
         str = left.toString( ) + ", ";
      str += data;
      if (right != null)
         str += ", "+right.toString( );
      return str;
   }
}
```

```
public class Test {

   public static final int[ ]
          values = {5, 1, 7, 15, 11, 17};

   public static void main(String[ ] args) {
      Node root = new Node(10);
      for (int i = 0; i < values.length; i++) {
         root.insertNode(values[i]);
      }
      System.out.println(root); // S1
      root.left = null; // S2
      root.right = null;
   }
}
```

The tree immediately before executing statement S1 looks like:



Q13. Immediately after S1 executes, what nodes in the tree are garbage? Give the numbers contained in the nodes that are garbage, or answer "none" if none of the nodes are garbage.

Q14. Immediately after S2 executes, what nodes in the tree are garbage? Give the numbers contained in the nodes that are garbage, or answer "none" if none of the nodes are garbage.

This program has questions 15 - 18. If something is printed by a line that is a question (has a Qx comment, where "x" is a number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist when answering other questions. If the statement prints nothing but is correct, answer "Ok". If a value is unitialized, answer "uninit".

```
public class Pair <K, V> {

   private K key;
   private V value;

   public Pair(K _key, V val) {
      key = _key;
      value = val;
   }

   public boolean equals(Pair<K,V> other) {
      if (key.equals(other.key)) {
         return (value.equals(other.value));
      } else return false;
   }

   public String toString( ) {
      String str = "(" + key.toString( );
      str += ", "+value.toString( )+")";
      return str;
   }
}

public class Duble {

   private double value;

   public Duble(double val) {
      value = val;
   }

   public boolean equals(Duble other) {
      return value == other.value;
   }

   public String toString( ) {
      return ""+value;
   }
}
```

```
public class Int {

   private int value;

   public Int(int val) {
      value = val;
   }

   public boolean equals(Int other) {
      return value == other.value;
   }

   public String toString( ) {
      return ""+value;
   }
}

public class Test {

   public static void main(String[ ] args) {
      Pair<String, Int> pair1 =
         new Pair("0", new Int(0));
      Pair<String, Int> pair2 =
         new Pair("1", new Int(1));
      Pair<String, Duble> pair3 =
         new Pair("1.0", new Duble(1.0));
      System.out.println(pair1); // Q15
      System.out.println(pair3); // Q16
      System.out.println(
         pair1.equals(pair2)); // Q17
   }
}
```

Q18: How many distinct Pair .class files (classes) exist in total to implement both the Pair generic class for "Pair<String, Int>" and "Pair<String, Duble>"?

This program has questions 19 to 32. If something is printed by a line that is a question say what is printed. If there is a either compile or runtime error, answer "Err" and assume the statement doesn't exist. If the statement prints nothing but is correct, answer "Ok". If a value is unitialized, answer "uninit".

```
public abstract class Base {

   public int value;

   public Base(int val) {
      value = val;
      System.out.println("Base");
   }

   public void f1( ) {
      System.out.println("B::f1");
   }

   public static void f2( ) {
      System.out.println("B::f2");
   }

   public void f3( ) {
      System.out.println("B::f3");
   }

   public void f4( ) {
      System.out.println("B::f4");
      f5( );
   }

   private void f5( ) {
      System.out.println("B::f5");
   }

   public String toString( ) {
      return "B::"+value;
   }
}
public class Derived extends Base {

   public Derived(int val) {
      super(val);
      System.out.println("Der");
   }

   public void f1( ) {
      System.out.println("D::f1");
   }

   public static void f2( ) {
      System.out.println("D::f2");
   }

   // Continuation of Derived.java
   private void f5( ) {
      System.out.println("D::f5");
   }

   public void f6( ) {
      System.out.println("D::f6");
   }
}
public class Test {

   private static void xchange(Object o1,
                               Object o2) {
      Object tmp = o1;
      o1 = o2;
      o2 = tmp;
      System.out.println(o1+" "+o2); // Q19
   }

   private static void change(Base o1) {
      o1.value = 5;
      o1 = null;
   }

   public static void main(String[ ] args) {
      Base b0 = new Base(0); // Q20
      Base b1 = new Derived(1); // Q21
      Base b2 = new Derived(2);

      xchange(b1, b2);
      System.out.println(b1+" "+b2); // Q22

      change(b1);
      System.out.println(b1); // Q23


      b1.f1( ); // Q24
      b1.f2( ); // Q25
      b1.f3( ); // Q26
      b1.f4( ); // Q27
      b1.f5( ); // Q28
      b1.f6( ); // Q29

      Derived d1 = b1; // Q30
      Derived d2 = (Derived) b2;
      d2.f6( ); // Q31
      d2.f5( ); // Q32
   }
}
```

This page intentionally left almost blank

This page and the next have questions 33 - 35.

Program 1

```
public interface Subject {
   public abstract void register(Observer o);
   public abstract void remove(Observer o);
}
import java.util.List;
import java.util.ArrayList;

public class Buss implements Subject {
   public Buss( ) {
     observers = new ArrayList<Observer>( );
   }

   public void register(Observer o) {
      observers.add(o);
   }

   public void remove(Observer o) {
      observers.remove(o);
   }

   public void notify(int i)  {
      for (Observer o : observers) {
         o.update(i);
      }
   }

   public void putDataOnBuss(int i) {
      notify(i);
   }

   private List<Observer> observers;
}

public interface Observer {
   public void update(int i);
}
public class OutputPort implements Observer {

   public OutputPort(int i) {
     portId = i;
   }

   public void update(int i) {
      System.out.print("Output port "+portId);
      System.out.println(" receives "+i);
   }

   private int portId = -1;
}
```

```
public class Test {

   private static Buss buss;
   private static void writeData(int low, int high)
   {
      for (int i = low; i < high; i++) {
         buss.putDataOnBuss(i);
      }
   }

   public static void main(String[ ] args) {
      buss = new Buss( );
      OutputPort port1 = new OutputPort(1);
      OutputPort port2 = new OutputPort(2);
      buss.register(port1);
      buss.register(port2);

      writeData(10, 12);

      buss.remove(port2);
      writeData(8, 10);

      buss.register(new OutputPort(3));
      writeData(6, 8);

   }
}
```

Program 2

```
import java.util.List;
import java.util.ArrayList;

public class Buss {
   public Buss(int[ ] portIds) {
      ports = new OutputPort[portIds.length];
      for (int i = 0; i < ports.length; i++) {
         ports[i] = new OutputPort(i);
      }
   }

   public void notify(int i)  {
      for (OutputPort p : ports) {
         p.write(i);
      }
   }

   public void putDataOnBuss(int i) {
      notify(i);
   }

   private OutputPort[ ] ports = null;
}
```

```
public class OutputPort {

   public OutputPort(int i) {
     portId = i;
   }

   public void write(int i) {
      System.out.print("Output port "+portId)
      System.out.println(" receives "+i);
   }

   private int portId = -1;
}
public class Test {
   public static void main(String[ ] args) {
      int[ ] portIds ={1, 3, 2};
      Buss buss = new Buss(portIds);

      buss.putDataOnBuss(6);
      buss.putDataOnBuss(7);

   }
}
```

Using Program 1 on the previous page, and the program on this page (Program 2) answer the following questions.

Q33. If we are writing a simulator for a "hot swap" buss where different output ports are being plugged into the buss, is Program 1 or Program 2 better? Take into account functionality, isolation of classes from change and simplicity of the code.

Q34. If we are writing a simulator for a buss where cards are plugged into the buss when it is created and never changed, is Program 1 or Program 2 better? Take into account functionality, isolation of classes from change and simplicity of the code.

Q35. What is the name of the pattern used in Program 1?

This page questions 36 and 37.

Program 3

```
public abstract class Disk {
   public Disk(String type) {
      diskType = type;
   }

   public String type( ) {
      return diskType;
   }

   private String diskType;
}
public class SSDDrive extends Disk {
   public SSDDrive( ) {
      super("SSD");
   }
}
public class SpinningDrive extends Disk {
   public SpinningDrive( ) {
      super("Spinning");
   }
}
public class Computer {
   public Computer(Disk d) {
       disk = d;
   }

   public void changeDisk(Disk d) {
      disk = d;
   }

   public String readDisk(int  addr) {
      return "reading addr "+addr+" on "+disk.type( ) + " disk";
   }

   private Disk disk = null;
}
```

```
public class ComputerBuilder {
   public static Computer build(String driveType) {

      Computer c = null;
      if (driveType.equals("ssd")) {
            c = new Computer(new SSDDrive( ));
      } else if (driveType.equals("spin")) {
            c = new Computer(new SpinningDrive( ));
      } else {
         assert false : "inv type: " + driveType;
      }
      return c;
   }
}
public class Main {
   public static void main(String[ ] args) {
      for (int i = 0; i < args.length; i++) {
         Computer c = ComputerBuilder.build(args[i]);
         System.out.println(c.readDisk(i));
      }
   }
}
```

Use Program 3 to answer the following questions:

Q36. Write the letter(s) corresponding to classes that would need to be changed if a new type of disk needs to be added to the prgoram.

A. Disk
B. SSDDrive
C. SpinningDrive
D. Computer
E. ComputerBuilder
F. Main

Q37. What pattern is used in Program 3?

This program has questions 38 and 39.

```
public abstract class Disk {
   public Disk(String type) {
      diskType = type;
   }

   public String type( ) {
      return diskType;
   }

   public void write(int track, int sector, int data) {
      System.out.print("writing "+data);
      System.out.println(" to ("+track+", "+sector+")");
   }

   private String diskType;
}
public class SSDDrive extends Disk {
   public SSDDrive(String name) {
      super(name);
   }
}
public class SpinningDrive extends Disk {
   public SpinningDrive(String name) {
      super(name);
   }
}
```

```
public class DiskBuilder {
   public static Disk build(String driveType) {

      Class<?> cls = null;
      Constructor<?> constructor = null;
      Disk disk = null;

      try {
         cls = Class.forName(driveType);
         constructor = cls.getConstructor(
            String.class);
         disk = (Disk) constructor.newInstance(
            driveType);  // S2
      } catch (Exception e) {
         e.printStackTrace( );
      }
      return disk;
   }
}
public class Main {
   public static void main(String[ ] args) {
      Disk disk =
         DiskBuilder.build("SSDDrive"); // S1
      disk.write(1, 2, 3);
      disk = DiskBuilder.build("SpinningDrive");
      disk.write(4, 5, 6);
      disk = DiskBuilder.build("Foo"); // Q38
      disk.write(7, 8, 9);
   }
}
```

Q39: When S1 in main is executed, what class's constructor is called by S2 in Diskbuilder.build?