COE318 Final Study Guide (Nov 10, 2014)

Questions

1. The following code will not compile. Find two ways to fix it so it will compile. What is the output once the problem is fixed?

```
public class Resistor {

   private final double r;
   private static int next = 1;
   private final int id;

   public Resistor(double r) throws BadRException {
      if (r == 0.0) {
         throw new BadRException("Cannot be Zero");
      }
      if (r < 0) {
         throw new BadRException("Cannot be negative");
      }
      this.r = r;
      id = next++;
   }

   @Override
   public String toString() {
      return "R" + id + " " + r + " ohms";
   }

   public static void main(String[] args) {
      Resistor p;

      p = new Resistor(5);
      System.out.println("" + p);
      p = new Resistor(0);

   }
}
```

2. What is the output when the following is run?

```
public class Resistor {

   private final double r;
   private static int next = 1;
   private final int id;

   public Resistor(double r) throws BadRException {
      if (r == 0.0) {
         throw new BadRException("Cannot be Zero");
      }
      if (r < 0) {
         throw new BadRException("Cannot be negative");
      }
      this.r = r;
      id = next++;
   }

   @Override
   public String toString() {
      return "R" + id + " " + r + " ohms";
   }

   public static void main(String[] args) {
      Resistor p;
      int i = 1;
      int j = 1;
      try {
         p = new Resistor(5);
         i++;
         System.out.println("" + p);
      } catch (BadRException ex) {
         System.out.println(j++);

      } finally {
         System.out.println(i);
      }
      try {
         p = new Resistor(-1);
         i++;
         System.out.println("" + p);
      } catch (BadRException ex) {
         System.out.println(j++);
      } finally {
         System.out.println(i);
      }
      try {
         p = new Resistor(2);
         i++;
         System.out.println("" + p);
      } catch (BadRException ex) {
         System.out.println(j++);

      } finally {
         System.out.println(i);
      }
      try {
         p = new Resistor(0.0);
         i++;
         System.out.println("" + p);
      } catch (BadRException ex) {
         System.out.println(j++);
      } finally {
         System.out.println(i);
      }
   }
}
```

3. What is the output when the following is run?

```
   import java.util.ArrayList;

   public class M {

      private int i;
      private static ArrayList<Integer> nums =
              new ArrayList<Integer>();

      public M(int i) {
         if (nums.contains(i)) {
            throw new IllegalArgumentException("Duplicates not
allowed");
         }
         nums.add(i);
         System.out.println(i);
         this.i = i;
      }

      public static void main(String[] args) {
         try {
```

```
         new M(3);
         new M(1);
         new M(4);
         new M(1);
         new M(5);
      } catch (Exception ex) {
      } finally {
         System.out.println(nums.size());
      }
   }
}
```

4. What is the output when the following is run?

```
import java.util.ArrayList;

public class N {
   private final ArrayList<String> names;

   public N() {
      names = new ArrayList<String>();
   }

   public void add(String s) {
      names.add(s);
   }

   public String[] getNamesArray() {
      return names.toArray(new String[0]);
   }

   public ArrayList<String> getArrayList() {
      return names;
   }

   @Override
   public String toString() {
      String s = "";
      for(String n : names) {
         s += n + "\n";
      }
      return s;
   }
   public static void main(String[] args) {
      N x = new N();
      x.add("Alice");
      x.add("Bob");
      String[] s = x.getNamesArray();
      s[0] = "Frodo";
      System.out.println(x);
      ArrayList n = x.getArrayList();
      n.set(0, "Bilbo");
      System.out.println(x);
   }
}
```

5. What is the output when the following is executed?

```
public class F {

   public static void main(String[] args) {
      int[] a = {2, 1, 7, 1, 8};
      System.out.println("a.length: " + a.length
              + " a[1]: " + a[1]);
      f(a);
      System.out.println("a.length: " + a.length
              + " a[1]: " + a[1]);
   }

   static void f(int[] b) {
      b[1] = 5;
      int[] c = {1, 2, 3};
      b = c;
   }
}
```

6. What is the output from the following program:

```
public class BadIException extends Exception {

    public BadIException() {
    }
    public BadIException(String msg) {
        super(msg);
    }
}

public interface I2 {
    void foo(int i) throws BadIException;
}

public abstract class A implements I2 {

    private int i;

    public A(int i) {
        this.i = i;
    }

    public void increment() {
        ++i;
    }

    public int getI() {
        return i;
    }

}
public class B extends A {

    public B(int j) {
        super(j);
        increment();
    }

    @Override
    public void foo(int i) throws BadIException {
        if(i <= 0) {
            throw new BadIException();
        }
        for (int j = 0; j < i; j++) {
            increment();
        }
    }

}
```

```
  public class C extends B {

      public C(int k) {
          super(k);
          increment();
      }

      @Override
      public void foo(int w) throws BadIException {
          super.foo(w);
          increment();
      }

      public static void main(String[] args) throws BadIException
{
          A b, c;
          I2 a;
          b = new B(2);
          b.foo(2);
          System.out.println("b.getI(): " + b.getI());
          c = new C(2);
          c.foo(2);
          System.out.println("c.getI(): " + c.getI());

          try {
              a = new C(3);
              a.foo(2);
              System.out.println("a.getI(): " + ((C) a).getI());
              B x = new B(5);
              x.foo(-2);
              System.out.println("Hello");
          } catch (BadIException ex) {
              System.out.println("Got BadIException");
          } finally {
              System.out.println("Goodbye");
          }
      }

  }
```

7. Consider the following code:

```
public class X extends Object {

    @Override
    public String toString() {
        return this.getClass().getName() + ":";
    }
}

class Y extends X {

    private int value;

    @Override
    public String toString() {
        return super.toString() + value;
    }

    public static void main(String[] args) {
        X yy = new Y();
        System.out.println("y:" + yy);
    }
}
```

a) What is the output when running Y's main method?

b) Would the output change if yy were declared as Y instead of X?

c) Would the output change if X were declared abstract?

8. What is the output from the following:

```
public interface IA {
    int cube(int b);//returns the cube of b
}

public abstract class Person implements IA {

    private final String name;
    private final int height;

    public Person(String name, int h) {
        this.name = name;
        height = h;
    }

    public abstract String getGender();

    @Override
    public int cube(int x) {
        return x * x * x;
    }

    @Override
    public String toString() {
        return name + ":" + getGender();
    }

    public int getHeight() {
        return height;
    }

    public String getName() {
        return name;
    }
}

class Male extends Person {

    public Male(String name, int h) {
        super(name, h);
    }

    @Override
    public String getGender() {
        return "M";
    }
}

class Female extends Person {

    public Female(String name, int h) {
        super(name, h);
    }

    @Override
    public String getGender() {
        return "F";
    }
```

```
     public static void main(String[] args) {
          Person mary = new Female("Mary", 2);
          System.out.println("mary:" + mary +
mary.cube(mary.getHeight()));
          System.out.println("Object?" + (mary instanceof
Object));
          System.out.println("Person?" + (mary instanceof
Person));
          System.out.println("IA?" + (mary instanceof IA));
          System.out.println("Male?" + (mary instanceof Male));
          System.out.println("Female?" + (mary instanceof
Female));
      }
  }
```

Answers

1. Method 1: Add a “throws” clause to main as follows:

```
 public class Resistor {

    private final double r;
    private static int next = 1;
    private final int id;

    public Resistor(double r) throws BadRException {
       if (r == 0.0) {
          throw new BadRException("Cannot be Zero");
       }
       if (r < 0) {
          throw new BadRException("Cannot be negative");
       }
       this.r = r;
       id = next++;
    }

    @Override
    public String toString() {
       return "R" + id + " " + r + " ohms";
```

```
     }

     public static void main(String[] args) throws BadRException
{
        Resistor p;

        p = new Resistor(5);
        System.out.println("" + p);
        p = new Resistor(0);

     }
  }
```

Method 2: Use “try-catch” as follows:

```
  public class Resistor {

     private final double r;
     private static int next = 1;
     private final int id;

     public Resistor(double r) throws BadRException {
        if (r == 0.0) {
           throw new BadRException("Cannot be Zero");
        }
        if (r < 0) {
           throw new BadRException("Cannot be negative");
        }
        this.r = r;
        id = next++;
     }

     @Override
     public String toString() {
        return "R" + id + " " + r + " ohms";
     }

     public static void main(String[] args) {
        Resistor p;
        try {
           p = new Resistor(5);
           System.out.println("" + p);
           p = new Resistor(0);
        } catch (BadRException ex) {
            }

        }
    }
```

In both cases, the output is:

```
R1 5.0 ohms
```

2. The output is:

```
R1 5.0 ohms
2
1
2
R2 2.0 ohms
3
2
3
```

3. The output is:

```
3
1
4
3
```

4. The output is:

```
Alice
Bob

Bilbo
Bob
```

5. The output is:

```
a.length: 5 a[1]: 1
a.length: 5 a[1]: 5
```

6. The output is:
7. a) y:Y:0 b) No c) No
8. The output is:

```
mary:Mary:F8
Object?true
Person?true
```

```
IA?true
Male?false
Female?true
```