**CS211 Fall 2014**

Dr. Kinga Dobolyi

**Exam 2 – Version A**

**DO NOT START**

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**public class Writer{**

**public static int count = -1;**

**private int size;**

**public Writer(){**

**size = 33;**

**}**

**public Writer(int size, int c){**

**size = size;**

**count = count + c;**

**}**

**public String toString(){**

**return "writer\n" + size;**

**}**

**}**

**import java.util.ArrayList;**

**public class Pen extends Writer{**

**private int color = 7;**

**public Pen(int c, int q, int r){**

**super(q, r);**

**color = c;**

**count++;**

**}**

**public Pen(int c){**

**Writer writer = new Writer(0,0);**

**color = c;**

**count++;**

**}**

**public String toString(){**

**System.out.println(super.toString());**

**return "pen \n" + color;**

**}**

**public static void main(String[] args){**

**Writer w1 = new Writer(4,5);**

**Pen p1 = new Pen(6,7,8);**

**Writer w3 = (Writer) p1;**

**Pen p2 = new Pen(15);**

**ArrayList list = new ArrayList();**

**System.out.println(count);**

**list.add(w1);**

**list.add(p1);**

**list.add(p2);**

**list.add(w3);**

**for (int i = 0; i < list.size(); i++)**

**System.out.println(list.get(i).toString());**

**}**

**}**

scratch paper and output

***SOME LINES MAY BE BLANK!!***

1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

12\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

16\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

17\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Provide the output of the code above. ***SOME LINES MAY BE BLANK!!***

1. a. 4 b. 24 c. 5 d. 0 e. 14

2. a. writer b. pen c. Object@A100

3. a.33 b. 4 c. -1 d. 0 e. 6

4. a. writer b. pen c. Object@A100

5. a. 7 b. 4 c. -1 d. 0 e. 6

6. a. writer b. pen c. Object@A100

7. a. 33 b. 4 c. -1 d. 0 e. 6

8. a. writer b. pen c. Object@A100

9. a. 33 b. 4 c. -1 d. 0 e. 6

10. a. writer b. pen c. Object@A100

11. a. 33 b. 15 c. -1 d. 0 e. 6

12. a. writer b. pen c. Object@A100

13. a. 7 b. 15 c. -1 d. 0 e. 6

14. a. writer b. pen c. Object@A100 d.

15. a. 33 b. 4 c. -1 d. 0 e. 6 d.

16. a. writer b. pen c. Object@A100 d.

17. a. 33 b. 4 c. -1 d. 0 e. 6 d.

Provide the output for the following code:

**public class Exam2{**

**public static void foo(int x){**

**if (x > 0 && x < 3){**

**String sam = null;**

**sam.toString();**

**}**

**try{**

**int[] arr = {4, 5, 6};**

**x = arr[x];**

**x = x / x;**

**}catch(NullPointerException e){**

**System.out.println("null");**

**}catch(IndexOutOfBoundsException e){**

**//do nothing**

**}finally{**

**System.out.println("bye");**

**}**

**}**

**public static void main(String[] args){**

**int[] nums = {7, 0, 1};**

**for (int i = 0; i < nums.length; i++){**

**try{**

**foo(nums[i]);**

**System.out.println("success");**

**}catch(NullPointerException e){**

**System.out.println("NPE");**

**}catch(Exception e){**

**System.out.println("main");**

**}**

18. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

19. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

20. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

21. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

22. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**}**

**}**

**}**

18. a. bye b. success c. main d. null e. NPE

19. a. bye b. success c. main d. null e. NPE

20. a. bye b. success c. main d. null e. NPE

21. a. bye b. success c. main d. null e. NPE

22. a. bye b. success c. main d. null e. NPE

Answer the following questions (***each are independent from one another***):

23. Imagine the type **Box** extends the type **Shape**. What must be true for the **Box** type?

a. **Box** must be a concrete class

b. If **Shape** is a class, **Box** can call **Shape**’s constructor(s) through the **super** keyword

c. A and B

24. If you see the code **Box.name** in a method of a class other than **Box**, what must be true about **name**?

a. It must be public

b. It must be static

c. A and B

d. None of the above

25. If you see the code

**Box b1 = new Box();**

**b1.foo();**

in a method of a class other than **Box**, what must be true about **foo**?

a. It must be public

b. It must be static

c. A and B

d. None of the above

26. Imagine I have a class **Box** that implements the **Comparable** interface. What must be true about **Box**?

a. It must be an abstract class if it does not have a **compareTo** method.

b. Writing code for the method with signature **public int compareTo(Box b)** implements the **Comparable** interface.

c. **Box** cannot implements any other interfaces.

d. A and B

e. A and C

27. Imagine I have a static method **foo** inside the class **Box**. What must be true about **foo**? Assume no methods are called on objects in **Box**.

a. **foo** can only call other static methods of **Box**

b. **foo** can only use static attributes of **Box**

c. A and B

d. None of the above.

28. Imagine I have the compiling code

**Box b = new Box(6);**

What must be true?

1. **Box** cannot have a default constructor.
2. **Box** must store an integer as an attribute.
3. **Box** cannot be an abstract class
4. A and C
5. B and C

29. Imagine a file called Box.java has the following code:

**public void foo();**

What cannot be true about **Box**?

1. **Box** is an interface
2. **Box** is an abstract class
3. **Box** is a concrete class
4. B and C

30. The following class declaration would compile (assume all imports exist and the rest of the class is properly defined between curly braces):

**public class Box implements Comparable, Iterator**

a. True b. False

31. An unchecked exception must be caught by a try-catch block, in order for Java to compile the code,

a. True b. False

32. If **Box** is a child class of **Shape**, the assignment **Box b1 = new Shape();** would compile.

a. True b. False

33. If **Box** is a child class of **Shape**, the assignment **Box b1 = (Box) new Shape();** would compile.

a. True b. False

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Extra scratch paper – TURN IN WITH YOUR EXAM