Name: _____

CS2113 Quiz 5

60 minutes. TAs will NOT be able to assist you with explaining the code below (this is part of the assessment).

1. Below we have a class hierarchy (child, parent, grandparent) of three simple classes. Trace through the code and give the output. [74 pts]

```
public class Bird{
	private int age;
	protected String science = "Aves";

	public Bird(int age){
		this.age = age;
		System.out.println("1");
	}

	public boolean canFly(int age){
		if (this.age > age)

	System.out.println("B");
		else

	System.out.println("C");
		return true;
	}

	public String toString(){
		System.out.println("2");
		return age + " " + science;
	}
}

public class Raptor extends Bird{
	private String food;

	public Raptor(){
		super(33);
		food = "mice";
		science = "Raptor";
		System.out.println("3");
	}

	public boolean canFly(){
		System.out.println("D");
		return false;
	}
}
```

```
public class Hawk extends Raptor{
	private String color;

	public Hawk(int age){
		super();
		color = "brown";
		System.out.println("4");
	}

	public boolean canFly(int age){
		System.out.println("E");
		return true;
	}

	public String toString(){
		System.out.println("5");
		return super.toString() + " "
			+ color;
	}
}
```

```
public class Driver{

        public static void main(String[] args){

                Bird b = new Bird(12);
                Raptor r = new Raptor();
                Hawk h = new Hawk(11);
                Bird x = new Hawk(13);
                Bird y = new Raptor();

                System.out.println(b.toString());
                System.out.println(r.toString());
                System.out.println(h.toString());
                System.out.println(x.toString());
                System.out.println(y.toString());

                System.out.println(b.canFly(5));
                System.out.println(r.canFly(6));
                System.out.println(r.canFly());
                System.out.println(h.canFly(7));
                System.out.println(h.canFly());
                System.out.println(x.canFly(8));
                System.out.println(y.canFly(9));

        }
}
```

OUTPUT:

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____
16. _____
17. _____
18. _____
19. _____
20. _____
21. _____
22. _____
23. _____
24. _____
25. _____
26. _____
27. _____
28. _____
29. _____
30. _____
31. _____
32. _____
33. _____
34. _____
35. _____
36. _____
37. _____

2. Explain why you would use an *abstract class* instead of a concrete class.

3. CIRCLE all true statements. You may justify your answers below:

    a. An abstract class must have at least one abstract method

    b. An abstract method has no method body

    c. An abstract class's constructor can only be called with `super(…)`

4. Complete the `addThing` method from the `ThingList` class below so that it would add a `Thing` to such a list. You must assume that no constructor is written (other than the default constructor Java always provides).

```
public class ThingList {

    private class Node{
        Node next = null;
        Thing data;

        public Node(Thing t){
            data = t;
        }
    }

    private Node head = null;

    public void addThing(Thing t){
        //your code here









    }
}
```

SCRATCH