1 Creating Cats

1.1 Given the Animal class, fill in the definition of the Cat class so that when `greet()` is called, the label “Cat” (instead of “Animal”) is printed to the screen. Assume that a Cat will make a “Meow!” noise if the cat is 5 years or older and “MEOW!” if the cat is less than 5 years old.

```java
public class Animal {
    protected String name, noise;
    protected int age;

    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }

    public String makeNoise() {
        if (age < 5) {
            return noise.toUpperCase();
        } else {
            return noise;
        }
    }

    public void greet() {
        System.out.println("Animal " + name + " says: " + makeNoise());
    }
}

public Cat(String name, int age) {
    super(name, age); // Call superclass' constructor.
    this.noise = "Meow!"; // Change the value of the field.
}

@Override
public void greet() {
    System.out.println("Cat " + name + " says: " + makeNoise());
}
```

2 Raining Cats and Dogs

2.1 Assume that Animal and Cat are defined as above. What would Java print on each of the indicated lines?

```java
public class TestAnimals {
    public static void main(String[] args) {
        Animal a = new Animal("Pluto", 10);
        Cat c = new Cat("Garfield", 6);
        Dog d = new Dog("Fido", 4);
        a.greet(); // (A) Animal Pluto says: Huh?
        c.greet(); // (B) Cat Garfield says: Meow!
        d.greet(); // (C) Dog Fido says: WOOF!
        a = c;
        ((Cat) a).greet(); // (D) Cat Garfield says: Meow!
        a.greet(); // (E) Cat Garfield says: Meow!
    }
}

public class Dog extends Animal {
    public Dog(String name, int age) {
        super(name, age);
        noise = "Woof!";
    }
    @Override
    public void greet() {
        System.out.println("Dog " + name + " says: " + makeNoise());
    }
    public void playFetch() {
        System.out.println("Fetch, " + name + "!");
    }
}
```

2.2 Consider what would happen if we added the following to the bottom of main under line 12:

```java
a = new Dog("Spot", 10);
d = a;
```

Why would this code produce a compiler error? How could we fix this error? This code produces a compiler error in the second line. The static type of d is Dog while the static type of a is Animal. Dog is a subclass of Animal, so this assignment will fail at compile time because not all Animals are Dogs. Use casting to address the problem.

```java
d = (Dog) a;
```

This represents a promise to the compiler that at runtime, a will be bound to an object that is compatible with the Dog type.
3 An Exercise in Inheritance Misery Extra

Cross out any lines that cause compile-time errors or cascading errors (failures that occur because of an error that happened earlier in the program), and put an X through runtime errors (if any). Don’t just limit your search to main, there could be errors in classes A, B, C. What does D.main output after removing these lines?

class A {
    public int x = 5;
    public void m1() {System.out.println("Am1-> " + x);} 
    public void m2() {System.out.println("Am2-> " + this.x);} 
    public void update() {x = 99;}
}
class B extends A {
    public void m2() {System.out.println("Bm2-> " + x);} 
    public void m2(int y) {System.out.println("Bm2y-> " + y);} 
    public void m3() {System.out.println("Bm3-> " + "called");}
}
class C extends B {
    public int y = x + 1;
    public void m2() {System.out.println("Cm2-> " + super.x);} 
    \ public void m4() {System.out.println("Cm4-> " + super.super.x); } can't do super.super 
    public void m5() {System.out.println("Cm5-> " + y);}
}
class D {
    public static void main (String[] args) {
        \ B a0 = new A(); Dynamic type must be B or subclass of B 
        \ a0.m1(); cascading: prev line failed, so a0 can't be initialized 
        \ a0.m2(16); cascading: prev line failed, so a0 can't be initialized 
        A b0 = new B();
        System.out.println(b0.x); [prints "5"]
        b0.m1(); [prints "Am1-> 5"]
        b0.m2(); [prints "Bm2-> 5"]
        \ b0.m2(61); m2 (int y) not defined in static type of b0 
        B b1 = new B();
        b1.m2(61); [prints "Bm2y-> 61"]
        b1.m3(); [prints "Bm3-> called"]
        A c0 = new C();
        c0.m2(); [prints "cm2-> 5"]
        \ C c1 = (A) new C(); Can't assign c1 to an A 
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m3(); [print Bm3-> called]
        \ c2.m4(); C.m4() is invalid 
        c2.m5(); [print Cm5-> 6]
        ((C) c0).m3(); [print Bm3-> called]
\( \text{(C) c0.m3(); NOT RUNTIME ERROR This would case the result of what the method returns and it returns void therefore compile-time error}
\)
\begin{verbatim}
b0.update();
b0.m1(); [print A\text{m1} \rightarrow 99]
\end{verbatim}