Nachdenkzettel: Vererbung

1. `class B extends X`. Jetzt fügen Sie eine neue Methode in `X` ein. Müssen Sie `B` anpassen?
2. Gegeben ist folgendes Code-Snippet:

   ```
   class B extends X {
       public void newMethodinB() { ... }
   }
   ```

   Jetzt fügen Sie eine neue `public` Methode in `B`, die abgeleitete Klasse, ein. Sie möchten diese neue Methode im Code verwenden. Prüfen Sie die folgenden Codezeilen:

   ```
   X x = new B();
   x.newMethodinB();
   ```

   Was stellen Sie fest? Welche Lösungen gibt es? Welche Lösung präferieren Sie? Warum?
3. Nehmen Sie an, `String` wäre in Java nicht `final`. Die Klasse `FileName` „`extends`“ die Klasse `String`. Ist das korrekt? Was kann passieren? Wie heißt das Prinzip dahinter? Tipp: Denken Sie daran, was mit Dateinamensregeln des Betriebssystems passiert. Beispielszenario:

   ```
   String s = new FileName();
   //...
   s.setValue(":datei?name.txt"); // Methodenname hypothetisch
   ```

1. Nein, außer die Methode abstrakt. In diesem Fall müsste die Methode aus X in B implementiert werden.
2.
    a. Die Methode kann nicht ausgeführt werden, weil das Objekt x als Objekt der Klasse X deklariert wird und die Methode in der Klasse X nicht existiert.
    b. Mögliche Lösungen sind:
        i. Ein Objekt der Klasse B erstellen, also: B b = new B()
        ii. Die Methode auch in X implementieren
        iii. x casten um ein Objekt der Klasse B zu bekommen ((B) x).newMethodInB()
    c. Ich würde die Lösung präferieren ein Objekt der Klasse B zu erstellen, weil man dadurch weder eine neue Methode implementieren noch ein Objekt casten muss.
3.
    a. Es ist korrekt, dass wenn String nicht final wäre, die Klasse FileName von String erben könnte.
    b. Strings sind immutable, also unveränderbar. Wenn aber die Klasse FileName Methoden hat, wodurch man Strings verändern könnte, dann würde die Unveränderlichkeit verletzt werden.
    c. Immutable Objects