Qual der Wahl, Wiederholung

```
public class A {
  public void f(Integer x) {
    System.out.println("Integer A");
  }
}

public class B extends A {
  public void f(Integer a) {
    System.out.println("Integer B");
  }
  public void f(int b) {
    System.out.println("int B");
  }
}
```

```
A a = new A();
B b = new B();
a.f(6);
b.f(6);
((A) b).f(6);
```

a.f(6) Output ist klar

b.f(6) die Klasse B überschreibt die Funktion f

((A) b) castet die Objektinstanz b zum Type A

Wir gehen nun davon aus, dass b vom Type A ist. In der Klasse A gibt es nur die Methode f(Integer). Somit können wir auch nur diese Methode aufrufen. Laut Java gibt es keine anderen Methoden, die aufgerufen werden können. Wegen Dynamic Binding / Dynamic Dispatch / Late Binding / Single Dispatch (zum Glück gibt es nur einen Namen...) wird die Methode f(Integer) in der Klasse B aufgerufen, da die diese Methode die Methode in A überschrieben hat. Das ist so permanent wie ein Blumen-Tattoo.

Anders ist es bei Feldern. Diese werden zwar auch vererbt. Allerdings können Felder nur versteckt werden. Konkret heisst das: Wenn die Tarndecke entfernt wird, sind die Felder wieder sichtbar.

Fazit: Beim Aufruf einer Methode ist der Type des Objektes massgeblich. Die Type der Variable ist relevant für die Bestimmung der Methoden, die aufgerufen werden können. Also es wird bestimmt welche Signaturen zugelassen werden. Der zu ausführende Code wird aber vom Objekt Type bestimmt

Lösung

```
Integer A
int B
Integer B
```