Practice writing equals functions

WE GIVE THE ANSWER TO QUESTION 3

In writing function equals, ask yourself:
(1) Did you implement the specification?
(2) Did you cast the parameter down to the appropriate class before attempting to reference fields?
(3) Did you write a return statement that returns a boolean?
(4) If you have to compare Strings, did you use the the function equals in class String, and not ==?
(5) Did you know that a static variable is not called a field?
   It is a "class variable".

1. Assume the following function equals is to be written in class Celeb. Complete its body.

```java
/** = "ob is a celebrity and has the same fields
   * as this Celeb object */
public boolean equals(Object ob) {
    if (ob instanceof Celeb) {
        Celeb b = (Celeb) ob;
        return (b.celebNum == this.celebNum)
                 && b.name.equals(this.name);
    } else { // ! (ob instanceof Celeb)
        return false;
    }
}
```

2. Assume the following function equals is to be written in class Pres. Complete its body.

```java
/** = "p is a Pres and has the same values in its fields
   * as this Pres object */
public boolean equals(Object p) {
    if (p instanceof Pres) {
        Pres pr = (Pres) p;
        return (pr.sYear == this.sYear)
                 && (pr.eYear == this.eYear)
                 && (pr.name.equals(this.name))
                 && (pr.prev == prev);
    } else {
        return false;
    }
```
3. Assume the following function equals is to be written in class Rational. Complete its body.

```java
/** = "r is of class Rational and has the same value as this Rational number" */
public boolean equals(Object r) {

    if (!r instanceof Rational) // The specification tells you
        return false; // that you need this. Read the spec!

    Rational rb = (Rational) r; // You need to check the
    // fields num and den. You can't do this using r
    // because its apparent class is Object. So cast r down
    // to Rational and store it in a Rational local variable.

    return num == rb.num && den == rb.den; // if you
    // don't write a return statement, it is wrong!! The
    // expression of the return statement is a simple
    // boolean expression that tests all the fields.
    // Make sure you read the specification of the function to make sure you test the right fields!!!!
}
```

4. Assume the following function equals is to be written in class Ath (defined below, along with class Celeb). Complete its body. This one has an added trickiness to it. How are you going to check that the names are the same when field name is private and there is not getter method for the name field? Answer, look in class Celeb and see whether anything can help.

```java
/** = "a is of class Ath and has the same name and sport as this Ath. */
public boolean equals(Object a) {

    if (a instanceof Ath) {
        Ath b = (Ath) a;
        return b.sport.equals(sport) &&
               super.equals(b);
    }

    return false;
}
```

```java
/** An instance maintains info about an athlete */
public class Ath extends Celeb {

    private String sport; // The athlete's sport
    private int win; // # of wins
    private int loss; // # of losses
    private double salary; // The athlete's salary

    ... methods are not shown ...
}
```

```java
/** Instance maintains info about a celebrity */
public class Celeb {

    private String name; // celebrity's name
    private String no; // celebrity's no. in the list of celebrities
    private int celebNum;

    // total number of celebrities
    private static int numCelebs = 0;

    /** = "ob is a celebrity and has the same fields as this Celeb object */
    public boolean equals(Object ob) { ... }

    ... other methods not shown ...
```