Creation des méthodes Hascode et Equals de bibliothèque redéfinie

```
public class EqualsBuilder
extends Object
implements Builder<Boolean>
```

Assists in implementing `Object.equals(Object)` methods.

This class provides methods to build a good equals method for any class. It follows rules laid out in Effective Java , by Joshua Bloch. In particular the rule for comparing `doubles`, `floats`, and arrays can be tricky. Also, making sure that `equals()` and `hashCode()` are consistent can be difficult.

Two Objects that compare as equals must generate the same hash code, but two Objects with the same hash code do not have to be equal.

All relevant fields should be included in the calculation of equals. Derived fields may be ignored. In particular, any field used in generating a hash code must be used in the equals method, and vice versa.

Typical use for the code is as follows:

```
public boolean equals(Object obj) {
  if (obj == null) { return false; }
  if (obj == this) { return true; }
  if (obj.getClass() != getClass()) {
    return false;
  }
  MyClass rhs = (MyClass) obj;
  return new EqualsBuilder()
              .appendSuper(super.equals(obj))
              .append(field1, rhs.field1)
              .append(field2, rhs.field2)
              .append(field3, rhs.field3)
              .isEquals();
}
```

```
public boolean equals(Pizza obj) {
    if (obj == null) {
        return false;
    }
    if (obj == this) {
        return true;
    }
    if (obj.getClass() != getClass()) {
        return false;
    }
    Pizza rhs = (Pizza) obj;
    return new EqualsBuilder().appendSuper(super.equals(obj)).append(code, rhs.code).append(nom, rhs.nom)
            .append(prix, rhs.prix).isEquals();
}
```