.0 round funtzioan:

```
private static double round(double d) {
// originala return Math.round(d * 100) / 100;
             return Math.round(d * 100) / 100.0;
}
```

Metodo honen funtzioa eskaintzen diogun aldagaia bi hamartarretan biribiltzea da. Hasieran geneukan aldagaia biribildu eta unitate osoa ematen zigun, hamartarrik gabe. Horri irtenbidea eman zaio .0 bat jartzea zatiketan.

Debuggerra erabili ondoren konturatu ginen Math.round-ek int bat ematen zigula eta guk Double bat behar genuen, hau lortzeko zatiketzaileari .0 bat ipini izan behar genuen Double bihurtzeko.

return -resto eskatuDirua funtzioan:

```
private static double eskatuDirua(double prezioTotala, AtomicBoolean ordainduta) {
             […]
// originala return -resto;
             return Math.abs(resto);
}
```

“Resto” aldagaiak beti zen negatiboa, eta metodo honekin positibora aldatu nahi genuen. Baina hasieran ez genuen lortzen.

Debuggerra erabili ondoren konturatu ginen Math.abs erabili behar zela ez zuelako zeinua aldatzen.

AtomicBoolean eskatuDirua funtzioan:

```
\\ originala private static double eskatuDirua(double prezioTotala, boolean
ordainduta) {
private static double eskatuDirua(double prezioTotala, AtomicBoolean ordainduta) {
                    […]
\\ originala              ordainduta = true
                          ordainduta.set(false);
                          return prezioTotala - resto;
                    }
             } while (resto > 0);
\\ originala ordainduta = true
             ordainduta.set(true);
             return Math.abs(resto);
}
```

Hasieran funtzio honetan erabiltzailea guztiz ordaindu duen ala ez jakiteko zenbaki positibo edo negatiboa erabiltzen genuen, baina konturatu ginen ezer ez ordaintzen zuenean edo zehatz ordaintzen zuenean 0 ematen zuela. Arazo hau konpontzeko boolean bat erabiltzea erabaki genuen.

Debuggerra erabiltzen konturatu ginen boolean aldagaia aldatzen genuenean ez zela aldatzen metodo honen kanpoan eta interneten bilatu eta gero aurkitu genuen hau egiteko AtomicBoolean edo array bat erabili ahal genuela boolean batekin. AtomicBoolean erabiltzea erabaki genuen argiagoa iruditzen zitzaigulako.