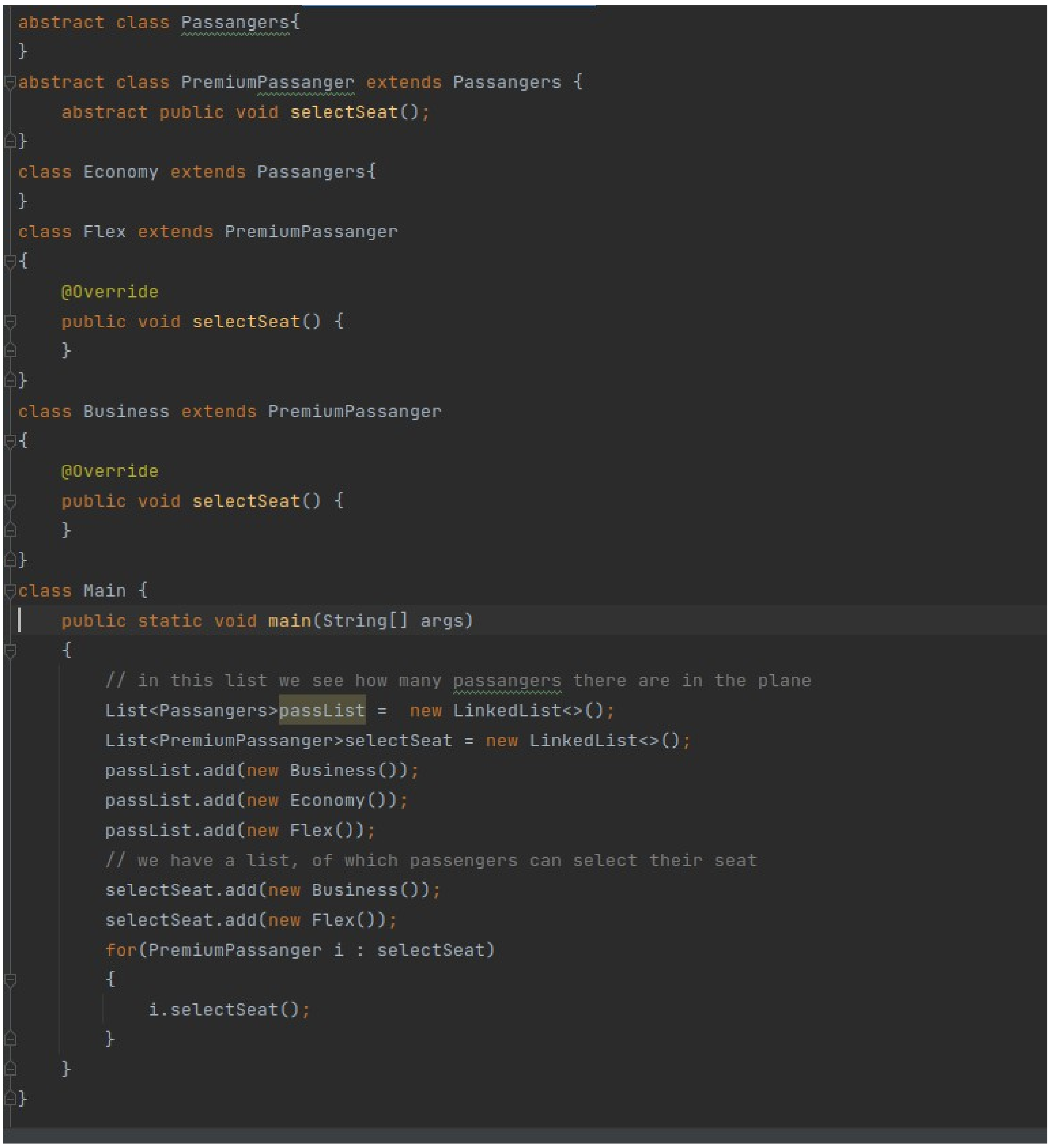
In the second example is the Liskov Substitution Principle broken, because each instance should behave as a passenger, but not everyone should have the ability to select a seat. I soled the problem so, that I added an extra class which is responsible, that the passengers can select the seats. In the source code I provided an example in which all passengers are listed, to know how many passengers there are and a specific list in which the passengers are listed, who can select in which seat they want to sit.



```java
abstract class Passangers{
}
abstract class PremiumPassanger extends Passangers {
    abstract public void selectSeat();
}
class Economy extends Passangers{
}
class Flex extends PremiumPassanger
{
    @Override
    public void selectSeat() {
    }
}
class Business extends PremiumPassanger
{
    @Override
    public void selectSeat() {
    }
}
class Main {
    public static void main(String[] args)
    {
        // in this list we see how many passangers there are in the plane
        List<Passangers>passList =  new LinkedList<>();
        List<PremiumPassanger>selectSeat = new LinkedList<>();
        passList.add(new Business());
        passList.add(new Economy());
        passList.add(new Flex());
        // we have a list, of which passengers can select their seat
        selectSeat.add(new Business());
        selectSeat.add(new Flex());
        for(PremiumPassanger i : selectSeat)
        {
            i.selectSeat();
        }
    }
}
```