```
            switch (currentTower) {
                //View tower selected, check money, add tower if enough
                case 1:
                    if (money >= Tower1.initCost(diff)) {
                        money = money - Tower1.initCost(diff);
                        updateMoney(moneyCounter);
                        towermap.addTower(x, y, tower: 1);
                        Tower1 newTower = new Tower1(x, y);
                        towerArray.add(newTower);
                    }
                    currentTower = 0;
                    break;
                case 2:
                    if (money >= Tower2.initCost(diff)) {
                        money = money - Tower2.initCost(diff);
                        updateMoney(moneyCounter);
                        towermap.addTower(x, y, tower: 2);
                        Tower2 newTower = new Tower2(x, y);
                        towerArray.add(newTower);
                    }
                    currentTower = 0;
                    break;
                case 3:
                    if (money >= Tower3.initCost(diff)) {
                        money = money - Tower3.initCost(diff);
                        updateMoney(moneyCounter);
                        towermap.addTower(x, y, tower: 3);
                        Tower3 newTower = new Tower3(x, y);
                        towerArray.add(newTower);
                    }
                    currentTower = 0;
                    break;
                default:
                    break;
            }
```

A team member wrote redundant code and also abused a switch statement in an attempt to write code that works and reduce development time. Doing this created technical debt, as the code is convoluted to read and modify since it does not follow object-oriented practices.

```
if (money >= TowerFactory.getTowerCost(currentTower, diff)) {
    money -= TowerFactory.getTowerCost(currentTower, diff);
    updateMoney(moneyCounter);
    towermap.addTower(x, y,  tower: 1);
    TowerInterface newTower = TowerFactory.getTower(currentTower, x, y);
    towerArray.add(newTower);
}
```

To fix the issue with this switch statement, I took advantage of interfaces and factories. I made it so that TowerInterface would require an initCost method so that I could create a spec/factory class that could generically refer to initCost() and instantiate different type of towers using the interface based on the current tower. This code is much shorter and much more understandable.