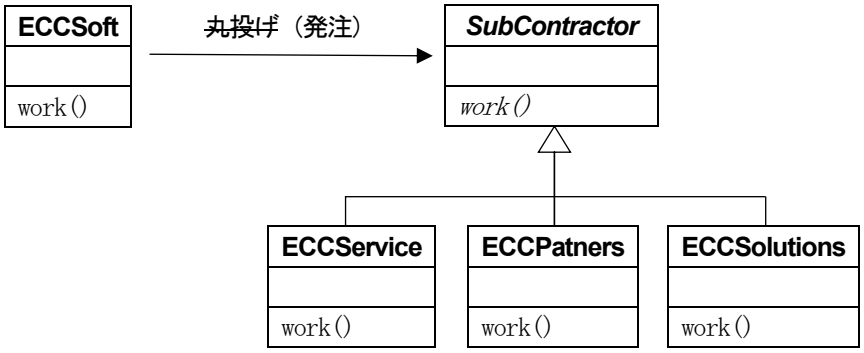
● ECC ソフトの憂鬱（Before）

「信頼と実績」の ECC ソフト株式会社では請け負った仕事は~~すべて下請け業者に丸投げしている。~~どうしても手が回らないときだけ協力会社に手伝ってもらっている。ただこのところどうもソフトウェアの品質が悪く、何度も「リファクタリング」されている。仕事を発注するたびに同じ協力会社だと思っていたのが、こっそり入れ替わっているようだ！協力会社は ECC サービス、ECC パートナーズ、ECC ソリューションズの 3 社。同じ会社に発注していたつもりが、毎回、同じ名前の違う会社に発注しているらしい。このままでは「信頼と実績」にキズが付く。協力会社が入れ替わっていないかどうか調査し、名前が同じであれば毎回、同じ会社（インスタンス）に発注するように業務改善せよ。

ECC ソフトと協力会社の関係



リスト 1 : ECC ソフトと協力会社の関係（ファイル「ECCSoft.java」）

```
// 元請け
public class ECCSoft {
     ⋮
    public void work() {
        SubContractor sub;
        switch((int)(Math.random() * 3)) {
            default:
            case 0: sub = new ECCService();     break;
            case 1: sub = new ECCPartners();    break;
            case 2: sub = new ECCSolutions();   break;
        }
        System.out.println(sub + "に丸投げします！");
        sub.work();                                    // 丸投げ
    }
}
```

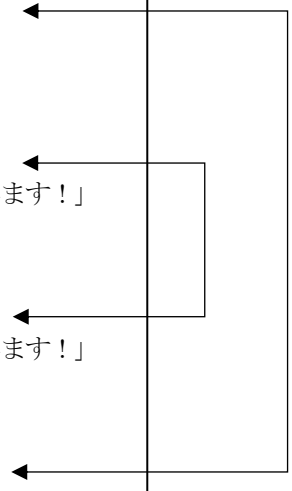
ECC ソフトでは乱数で決めた協力会社へ丸投げしている。

リスト１：ECC ソフトと協力会社の関係（続き）

```
// 下請け
abstract class SubContractor {
    private String name;
    public SubContractor(String name) { this.name = name; }
    public String toString() { return name; }
    public abstract void work();
}
// ECC サービス株式会社
class ECCService extends SubContractor {
    public ECCService() { super("ECC サービス株式会社"); }
    public void work() { System.out.println("ECC サービス株式会社「何とかがんばってみます！」"); }
}
// ECC パートナーズ株式会社
class ECCPartners extends SubContractor {
    public ECCPartners() { super("ECC パートナーズ株式会社"); }
    public void work() { System.out.println("ECC パートナーズ株式会社「下請けはつらいなー！」"); }
}
// ECC ソリューションズ株式会社
class ECCSolutions extends SubContractor {
    public ECCSolutions() { super("ECC ソリューションズ株式会社"); }
    public void work() {System.out.println("ECC ソリューションズ株式会社「よろこんでお引き受けいたします」");}
}
```

実行時の画面

```
信頼と実績の ECC ソフト株式会社です！
どんな課題でも私たちが真摯に解決します！！
どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC サービス株式会社に丸投げします！
ECC サービス株式会社「何とかがんばってみます！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC ソリューションズ株式会社に丸投げします！
ECC ソリューションズ株式会社「よろこんでお引き受けいたします！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC ソリューションズ株式会社に丸投げします！
ECC ソリューションズ株式会社「よろこんでお引き受けいたします！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC サービス株式会社に丸投げします！
ECC サービス株式会社「何とかがんばってみます！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC パートナーズ株式会社に丸投げします！
ECC パートナーズ株式会社「下請けはつらいなー！」
```



会社名は同じだが、どうも違う会社らしい。

● J2Kad28D「ID 番号の調査」※クラス変数とインスタンス変数の復習

協力会社のインスタンスを生成するたびに異なる ID 番号（会社間でも異なるようにする）を割り当てる処理を追加し、協力会社が入れ替わっていることを確認せよ。

SubContractor に追加するメンバ

追加するメンバ	説明
private static int count = 0;	生成した協力会社の数
private int myID;	ID 番号（コンストラクタで設定する）
public int getID();	ID 番号を返す。

SubContractor
count ： int myID ： int …
getID() ： int …

ECCSoft の修正

修正するメソッド	修正箇所
public void work()	「～株式会社に丸投げします！」に ID 番号の表示を追加する。 →「～株式会社（ID：#）に丸投げします！」（#はその会社の ID 番号）

課題完成の画面

```
信頼と実績の ECC ソフト株式会社です！
どんな課題でも私たちが真摯に解決します！！
どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC サービス株式会社（ID：0）に丸投げします！
ECC サービス株式会社「何とかがんばってみます！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC ソリューションズ株式会社（ID：1）に丸投げします！
ECC ソリューションズ株式会社「よろこんでお引き受けいたします！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC ソリューションズ株式会社（ID：2）に丸投げします！
ECC ソリューションズ株式会社「よろこんでお引き受けいたします！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC サービス株式会社（ID：3）に丸投げします！
ECC サービス株式会社「何とかがんばってみます！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC パートナーズ株式会社（ID：4）に丸投げします！
ECC パートナーズ株式会社「下請けはつらいなー！」
```

ID 番号が異なっている

● J2Kad28C「インスタンスはひとつだけ！（Singleton）」

協力会社のインスタンスが複数生成されないように（1 つの会社のインスタンスは 1 つだけになるように）処理を修正せよ。

ECCService に追加・修正するメンバ（ECCPartners と ECCSolutions も同様の追加・修正）

追加・修正するメンバ	説明
private static ECCService instance = new ECCService();	ECCService のインスタンス
public static ECCService getInstance()	instance を返す
private ECCService()	コンストラクタは private にする

ECCService
- instance …
+ getInstance() - ECCService() …

ECCSoft の修正

修正するメソッド	修正箇所
public void work()	協力会社のインスタンスを new している →協力会社のインスタンスを getInstance で取得する

課題完成の画面

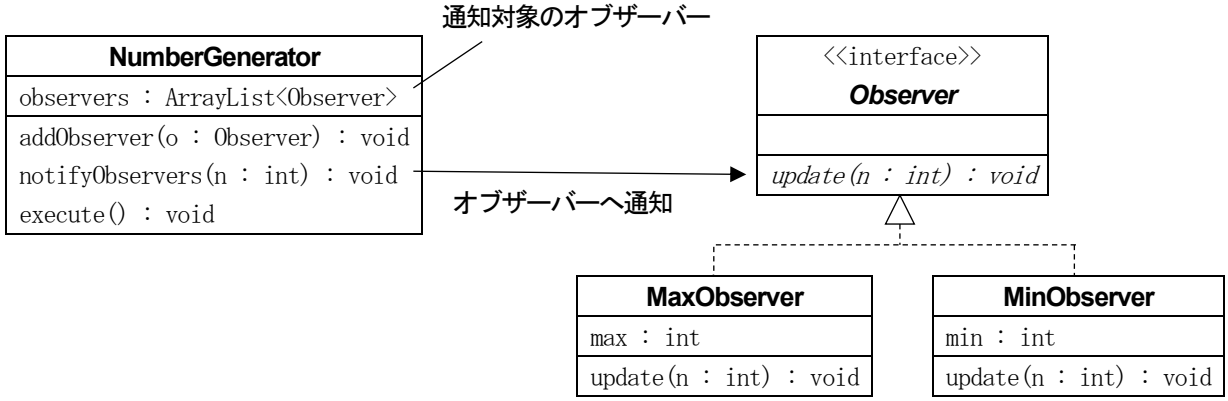
```
信頼と実績の ECC ソフト株式会社です！
どんな課題でも私たちが真摯に解決します！！
どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC サービス株式会社（ID：0）に丸投げします！
ECC サービス株式会社「何とかがんばってみます！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC ソリューションズ株式会社（ID：1）に丸投げします！
ECC ソリューションズ株式会社「よろこんでお引き受けいたします！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC ソリューションズ株式会社（ID：1）に丸投げします！
ECC ソリューションズ株式会社「よろこんでお引き受けいたします！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC サービス株式会社（ID：0）に丸投げします！
ECC サービス株式会社「何とかがんばってみます！」

どうしますか？（0：仕事を依頼する、-1：もういい）＞0
ECC パートナーズ株式会社（ID：2）に丸投げします！
ECC パートナーズ株式会社「下請けはつらいなー！」
```

会社が同じなら ID 番号も同じ

● J2Kad28B「最大値と最小値（Observer)」

乱数で 0 から 99 までの整数を発生させる NumberGenerator クラスが準備されている。発生した整数がこれまでで最大かどうかを判定する MaxObserver クラスを作成し、最大であれば「これまでの最大値です！」と表示する処理を作成せよ。同様にこれまでで最小かどうかを判定する MinObserver クラスも作成せよ。

課題完成時のクラス図（ファイル「NumberGenerator.java」に作成）



MaxObserver クラス（ファイル「NumberGenerator.java」に作成、MinObserver クラスも同様に作成する）

メンバ	説明
private int max = 0;	これまでの最大値
public void update(int n)	n が max より大きければ max を更新し、「MaxObserver：これまでの最大値です！」と表示

NumberGenerator に追加・修正するメンバ

追加・修正するメンバ	説明
private ArrayList<Observer> observers = new ArrayList<>();	通知対象のオブザーバー
public void addObserver(Observer o)	通知対象のオブザーバーo を追加する
public void notifyObservers(int n)	整数 n が発生したことを各オブザーバーに通知する
public void execute()	整数 n を生成したのち、オブザーバーへの通知（notifyObserbers）を追加する

main メソッドの仕様

① NumberGenerator に MaxObserver と MinObserver を追加し、実行（execute）する。

課題完成の画面

```
どうしますか？（0：generate、-1：やめる）>0
68 が出ました！
MaxObserver：これまでの最大値です！
MinObserver：これまでの最小値です！

どうしますか？（0：generate、-1：やめる）>0
34 が出ました！
MinObserver：これまでの最小値です！

どうしますか？（0：generate、-1：やめる）>0
```

（続き）

```
44 が出ました！

どうしますか？（0：generate、-1：やめる）>0
86 が出ました！
MaxObserver：これまでの最大値です！

どうしますか？（0：generate、-1：やめる）>0
62 が出ました！

どうしますか？（0：generate、-1：やめる）>-1
```

● J2Kad28A「平均値と素数判定」

J2Kad28B に以下のオブザーバーを追加せよ。

・AvrObserver　　これまで発生した整数の平均値を表示する。
・PrimeObserver　　発生した整数が素数かどうか判定する。

課題完成の画面

```
どうしますか？（0：generate、-1：やめる）＞0
41 が出ました！
MaxObserver：これまでの最大値です！
MinObserver：これまでの最小値です！
AvrObserver：これまでの平均は 41.0 です！
PrimeObserver：これは素数です！

どうしますか？（0：generate、-1：やめる）＞0
43 が出ました！
MaxObserver：これまでの最大値です！
AvrObserver：これまでの平均は 42.0 です！
PrimeObserver：これは素数です！

どうしますか？（0：generate、-1：やめる）＞0
71 が出ました！
MaxObserver：これまでの最大値です！
AvrObserver：これまでの平均は 51.666666666666664 です！
PrimeObserver：これは素数です！

どうしますか？（0：generate、-1：やめる）＞0
5 が出ました！
MinObserver：これまでの最小値です！
AvrObserver：これまでの平均は 40.0 です！
PrimeObserver：これは素数です！

どうしますか？（0：generate、-1：やめる）＞0
19 が出ました！
AvrObserver：これまでの平均は 35.8 です！
PrimeObserver：これは素数です！

どうしますか？（0：generate、-1：やめる）＞0
44 が出ました！
AvrObserver：これまでの平均は 37.166666666666664 です！

どうしますか？（0：generate、-1：やめる）＞-1
```

● J2Kad28S「FizzBuzz＋わん！（Observer 版）」

Observer パターンを使って「FizzBuzz＋わん！」を作成せよ。仕様は以下の通り。なお、すべてのクラス・インターフェイスはファイル「J2Kad28S.java」に作成すること（Observer インターフェイスは J2Kad28A で作成したものをそのまま実装しても OK。もし新規に作成する場合は、名前を変更して作成すること）。

FizzBuzz＋わん！の仕様

1 から 39 までの数字に対して以下の判定を行う。

① 3 の倍数のとき「Fizz」と表示
② 5 の倍数のとき「Buzz」と表示
③ ①でも②でのないとき数字をそのまま表示
④ 3 の倍数または 3 を含む数値（13 や 35 など）のとき「わん！」と表示

課題完成時の画面

```
1
2
Fizz わん！
4
Buzz
Fizz わん！
7
8
Fizz わん！
Buzz
11
Fizz わん！
13 わん！
14
FizzBuzz わん！
16
17
Fizz わん！
19
Buzz
Fizz わん！
22
23 わん！
Fizz わん！
Buzz
26
Fizz わん！
28
29
FizzBuzz わん！
31 わん！
32 わん！
Fizz わん！
34 わん！
Buzz わん！
Fizz わん！
37 わん！
38 わん！
Fizz わん！
```

● J2Kad28X「のび太がいっぱい！」

ECC が動画配信に参入した！その名も「ECC ビデオ」、豊富なコンテンツで視聴者を楽しませている。ところがどういうわけか同じ視聴者が複数人登録されている。同じ視聴者が複数回、登録申込みをするのが原因だが、複数回申込みしても一人しか登録しないように修正せよ。

課題完成前の画面（Before）

```
誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞0
のび太を登録しました！

誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞1
しずかを登録しました！

誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞0
のび太を登録しました！                                          ← 多重登録

誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞-1

＊ようこそ！豊富なコンテンツの ECC ビデオへ！！＊
何を見ますか？（0：ドラえもん、1：ポケモン、2：アンパンマン、-1：もうあきた）＞0
＊ドラえもんを配信します！＊
のび太：ぼく全部好きだよ！                                      ← のび太が２人
のび太：ぼく全部好きだよ！
しずか：これ一番見たかったの！
```

課題完成時の画面（After）※のび太以外も多重登録できないようにすること

```
誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞0
のび太を登録しました！

誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞1
しずかを登録しました！

誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞0
のび太はすでに登録されています！                                ← 多重登録できない

誰を登録しますか？（0：のび太、1：しずか、2：スネ夫、3：ジャイアン、-1：これでいい）＞-1

＊ようこそ！豊富なコンテンツの ECC ビデオへ！！＊
何を見ますか？（0：ドラえもん、1：ポケモン、2：アンパンマン、-1：もうあきた）＞0
＊ドラえもんを配信します！＊
のび太：ぼく全部好きだよ！                                      ← のび太は１人だけ
しずか：これ一番見たかったの！
```

ヒント①：そもそも「のび太」は一人しかいないはず。
ヒント②：ArrayList は同じインスタンスを複数登録できる。重複を防ぐには HashSet（実践編 P.106、または検索）を使う。