Contracts for objects

Inheritance

Liskov Substitution Principle

- If S is a declared subclass of T, then object of type S should behave as objects of type T if treated as objects of type T

Weakening Preconditions

```
class T {
    // require P
    // ensure Q
    void m() {…} }
class S extends T {
    // require P0
    // ensure Q
    @Override void m() {…} }
```

- We need that P implies P_0
- A client that treats an S object as a T object will ensure P prior to the call and hence P_0

Strengthening Postconditions

```
class T {
    // require P
    // ensure Q
    void m() {…} }
class S extends T {
    // require P
    // ensure Q0
    @Override void m() {…} }
```

- We need that Q_0 implies Q
- A client that treats an S object as a T object will expect Q after the call. If instead Q_0 is true, then that must imply that Q is true.

In general

```
class T {
    // require P
    // ensure Q
    void m() {…} }
class S extends T {
    // require P0
    // ensure Q0
    @Override void m() {…} }
```

- We need that $(P_0 \Rightarrow Q_0)$ implies $(P \Rightarrow Q)$

Framing

```
class T {
    // require P
    // modifies M
    // ensure Q
    void m() {…} }
class S extends T {
    // require P0
    // modifies M0
    // ensure Q0
    @Override void m() {…} }
```

- In general $M_0 \subseteq M$ except that M_0 may contain fields introduced in S.