Separating mechanism and policy

Everyday example of mechanism/policy separation

- the use of “card keys” to gain access to locked doors
  - The mechanisms do not impose any limitations on entrance policy (which people should be allowed to enter which doors, at which times).
  - These decisions are made by a centralized security server, which (in turn) probably makes its decisions by consulting a database of room access rules.
  - Specific authorization decisions can be changed by updating a room access database.
  - If the rule schema of that database proved too limiting, the entire security server could be replaced while leaving the fundamental mechanisms (readers, locks, and connections) unchanged.

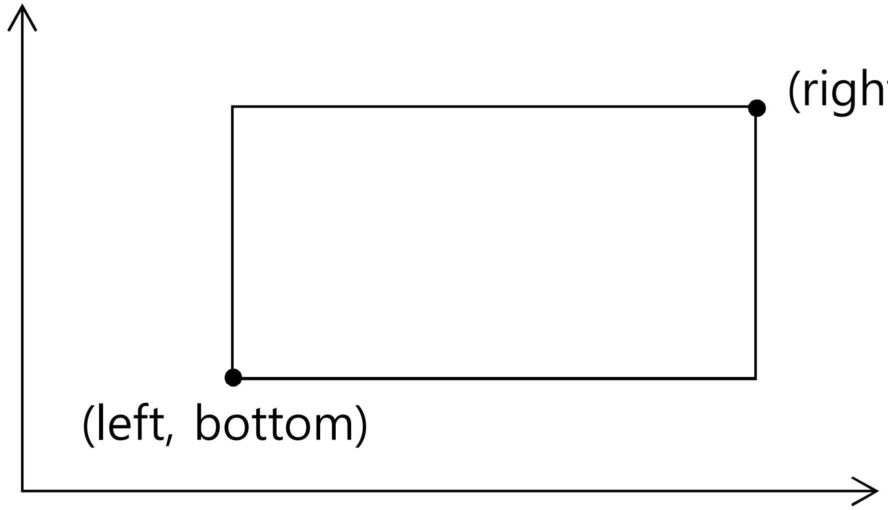
Everyday example of mechanism/policy separation

- Contrast this with issuing physical keys: if you want to change who can open a door, you have to issue new keys and change the lock.
- This intertwines the unlocking mechanisms with the access policies. For a hotel, this is significantly less effective than using key cards.

Separating mechanism from policy - function implementation

- Function that calculates area of rectangle

```
int rect_area(int left, int top, int right, int bottom) {
        return (right - left) * (top - bottom);
}
```



Adding some exception/error handling

```
int rect_area(int left, int top, int right, int bottom) {
        if (left >= right)
                left = right;
        if ( bottom >= top)
                bottom = top;
        return (right - left) * (top - bottom);
}
```

Another function is needed – returning error value if input rectangle is invalid

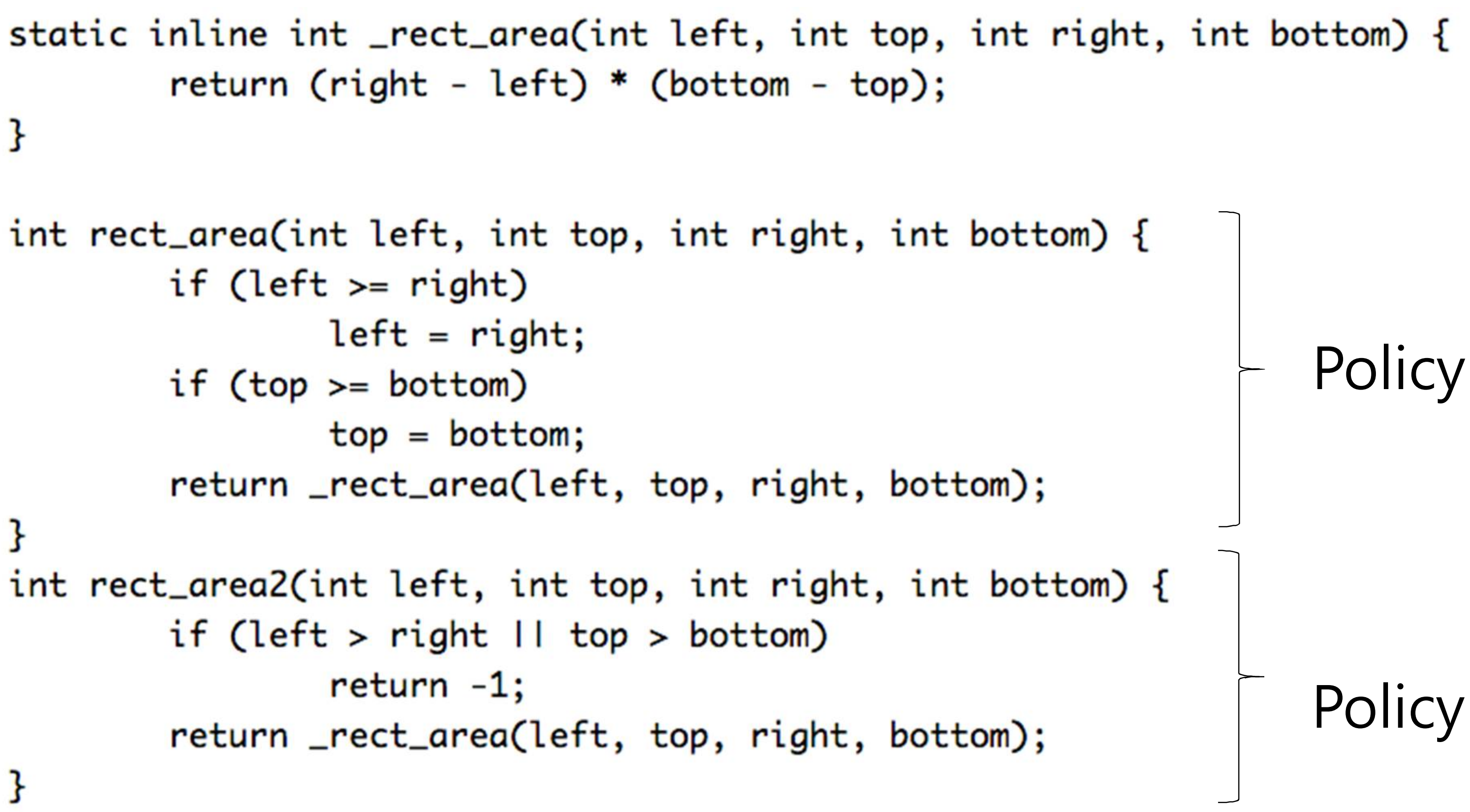
```
int rect_area2(int left, int top, int right, int bottom) {
        if (left > right || bottom > top)
                return -1;
        return (right - left) * (top – bottom);
}
```

rect_area() vs. rect_area2()

- Code for calculating area of rectangle is duplicated. Just one-line code. So, dulplicating is not a big deal. But, it’s not good in terms of code structure. Why did this happen?

- Calculating rectangle area is ‘Mechanism’. But, error handling is ‘Policy’ at this example.

Example should be implemented like this:



```
static inline int _rect_area(int left, int top, int right, int bottom) {
        return (right - left) * (bottom - top);
}

int rect_area(int left, int top, int right, int bottom) {
        if (left >= right)
                left = right;
        if (top >= bottom)
                top = bottom;
        return _rect_area(left, top, right, bottom);
}
int rect_area2(int left, int top, int right, int bottom) {
        if (left > right || top > bottom)
                return -1;
        return _rect_area(left, top, right, bottom);
}
```

Mechanism

Policy 1

Policy 2