Possible but rejected approaches to dynamic dispatch:

- Cast in the caller
  - check type field in every function that needs to call a derived method (e.g., calc_pts)
  - cast the base pointer to the right derived type
  - call the method

```
1  int pts0;
2  if ( p0->get_type() == PAWN ) {
3    Pawn* pawn_p = (Pawn*) p0;
4    pts0 = pawn_p->get_pts();
5  }
6  else if ( p0->get_type() == ROOK ) {
7    Rook* rook_p = (Rook*) p0;
8    pts0 = rook_p->get_pts();
9  }
```

  Works, but every caller must know all derived types and the logic duplicates everywhere.

- Cast in the base class (move the type-checking logic into a base class method)
  - (Piece::get_pts) checks its own type field
  - casts this to correct sub-type
  - calls the method

```
1  int get_pts() const
2  {
3    if ( m_type == PAWN ) {
4      Pawn* pawn_p = (Pawn*) this;
5      return pawn_p->get_pts();
6    }
7    else if ( m_type == ROOK ) {
8      Rook* rook_p = (Rook*) this;
9      return rook_p->get_pts();
10   }
11 }
```

  Centralizes the dispatch, but the base class now has to know about all derived classes