```
public class GameStateDiscard {
  private static int N_PILES = 4;
  private List<Card> _discards;
  private Pile[] _piles;

  /**
   * Discard all cards of the specified suit other than the highest one.
   * @param suit The suit to discard.
   */
  //-----------------------------------------------------------------
  public void discard0(Suit suit) {
    // Figure out the pile with the highest card of the specified suit
    int highest = -1;
    for (int pile = 0; pile < N_PILES; pile++) {
      Card current = _piles[pile].getTop();
      Card highCard = _piles[highest].getTop();
      if (current.getSuit() == suit) {
        if (highest == -1) {
          highest = pile;
        }
        if (current.compareRank(highCard) > 0) {
          highest = pile;
        }
      }
    }
    // We now know what pile the highest card of the specified suit is in.
    // Discard everything else
    for (int pile = 0; pile < N_PILES; pile++) {
      if (pile != highest && _piles[pile].getTop().getSuit() == suit) {
        _discards.add(_piles[pile].discard());
      }
    }
  }
```

```
// PTO PTO PTO PTO PTO PTO PTO PTO PTO PTO PTO PTO PTO PTO
```

```
  //------------------------------------------------------------------
  public void discard1(Suit suit) {
    int highest = -1;
    for (int pile = 0; pile < N_PILES; pile++) {
      Card current = _piles[pile].getTop();
      if (current.getSuit() == suit) {
        if (highest == -1) {
          highest = pile;
        } else {
          Card highCard = _piles[highest].getTop();
          if (current.compareRank(highCard) > 0) {
            highest = pile;
          }
        }
      }
    }
    for (int pile = 0; pile < N_PILES; pile++) {
      if (pile != highest && _piles[pile].getTop().getSuit() == suit) {
        _piles[pile].discard();
      }
    }
  }

  //------------------------------------------------------------------
  public void discard2(Suit suit) {
    int highest = -1;
    for (int pile = 0; pile < N_PILES; pile++) {
      Card current = _piles[pile].getTop();
      if (current.getSuit() == suit && (highest == -1 ||
        current.getRank().compareTo(_piles[highest].getTop().getRank()) > 0)) {
        highest = pile;
      }
    }
    for (int pile = 0; pile < N_PILES; pile++) {
      if (pile != highest && _piles[pile].getTop().getSuit() == suit) {
        _discards.add(_piles[pile].discard());
      }
    }
  }
}
```