```
public boolean removeCard(String name) {
    int newDeckCardsNum = currCard;
    currCard = 0;
    Card[] newDeck = new Card[maxCards];
    Card card = null;
    for (int i=0; i < newDeckCardsNum; i++) {
        if (deck[i].getName().equals(name)) {
            //System.out.println("Debug: Found card to remove " + name);
            card = deck[i];
        }
        else {
            //System.out.println("Debug: Keeping card " + deck[i].getName());
            newDeck[currCard] = deck[i];
            currCard += 1;
        }
    }
    deck = newDeck;
    if (card != null) {
        return true;
    }
    else {
        return false;
    }
}
```

Reduced code complexity by using a vector instead of an array:

```
public boolean removeCard(String name) {
    Card card = null;
    for (Card c : deck) {
        if (c.getName().equals(name)) {
            card = c;
        }
    }
    if (card != null) {
        deck.remove(card);
        return true;
    }
    else {
        return false;
    }
}
```

```
package edu.colorado.group18.battleship;

public abstract class Ability {}
```

-abstract class => interface----------

```
package edu.colorado.group18.battleship;

public interface Ability {}
```

##########

```
package edu.colorado.group18.battleship;

public class Sonar extends Ability {
```

-extends => implements----------

```
package edu.colorado.group18.battleship;

public class Sonar implements Ability {
```

IntelliJ Deodorant found a Long method code smell:

```
public void removeStrayShipCells() {
    ShipCell currCell = null;
    for (int y = 0; y < rows; y++) {
        for (int x = 0; x < cols; x++) {
            if (cells[y][x] instanceof ShipCell) {
                currCell = (ShipCell) cells[y][x];
                if (currCell.getShipRef().getPlaced() == false) {
                    cells[y][x] = new Cell();
                }
            }
        }
    }
}
```

Added a helper function to fix this:

```
public void removeStrayShipCells() {
    ShipCell currCell = null;
    for (int y = 0; y < rows; y++) {
        for (int x = 0; x < cols; x++) {
            checkCellHelper(y, x);
        }
    }
}

private void checkCellHelper(int y, int x) {
    ShipCell currCell = null;
    if (cells[y][x] instanceof ShipCell) {
        currCell = (ShipCell) cells[y][x];
        if (currCell.getShipRef().getPlaced() == false) {
            cells[y][x] = new Cell();
        }
    }
}
```