```
public void shipDestroyed(){
    for (int i = 0; i < shipList.size(); i++){
        boolean destroyed = true;
        for (int j = 0; j < shipList.get(i).getSize(); j++){
            //if
(!playground.getPlayground()[shipList.get(i).getPos()[0][j]][shipList.get(i
).getPos()[1][j]].getText().equals("X")){
            if
(!playground.getPlayground()[shipList.get(i).getPos()[0][j]][shipList.get(i
).getPos()[1][j]].getBackground().equals(Color.ORANGE)){
                    destroyed = false;
            }
        }
        if (destroyed){
            for (int j = 0; j < shipList.get(i).getSize(); j++){

playground.getPlayground()[shipList.get(i).getPos()[0][j]][shipList.get(i).
getPos()[1][j]].setBackground(Color.RED);
            }
            shipList.remove(i);
        }
    }
    for (int i = 0; i < playground.getPlayground().length; i++){
        for (int j = 0; j < playground.getPlayground()[i].length; j++){
            if(playground.hasNeighbor(i,j,Color.RED) &&
!playground.getPlayground()[i][j].getBackground().equals(Color.RED)){
                    playground.getPlayground()[i][j].setEnabled(false);

playground.getPlayground()[i][j].setBackground(Color.YELLOW);
            }
        }
    }
}
```

```
public void hit(int x, int y){
    for (Ship ship : shipList) {
        for (int j = 0; j < ship.getSize(); j++) {
            if (ship.getPos()[0][j] == x && ship.getPos()[1][j] == y) {

//playground.getPlayground()[ship.getPos()[0][j]][ship.getPos()[1][j]].setT
ext("X TREFFER");

playground.getPlayground()[ship.getPos()[0][j]][ship.getPos()[1][j]].setBac
kground(Color.ORANGE);
                shipDestroyed();
                return;
            }
            System.out.println();
        }
    }
    //playground.getPlayground()[x][y].setText("X KEIN TREFFER");
    playground.getPlayground()[x][y].setBackground(Color.WHITE);
}
```

```
public void disableNotPlaceable(int size) {
    size--;
    for (int i = 0; i < playground.length; i++) {
        for (int j = 0; j < playground[i].length; j++) {
            boolean placeable = false;
            try {
                if
(playground[i][j+size].getBackground().equals(waterColor) &&
playground[i][j].getBackground().equals(waterColor)) {
                    placeable = true;
                }
            } catch (ArrayIndexOutOfBoundsException ignored) {
            }
            try {
                if (playground[i][j-
size].getBackground().equals(waterColor) &&
playground[i][j].getBackground().equals(waterColor)) {
                    placeable = true;
                }
            } catch (ArrayIndexOutOfBoundsException ignored) {
            }
            try {
                if
(playground[i+size][j].getBackground().equals(waterColor) &&
playground[i][j].getBackground().equals(waterColor)) {
                    placeable = true;
                }
            } catch (ArrayIndexOutOfBoundsException ignored) {
            }
            try {
                if (playground[i-
size][j].getBackground().equals(waterColor) &&
playground[i][j].getBackground().equals(waterColor)) {
                    placeable = true;
                }
            } catch (ArrayIndexOutOfBoundsException ignored) {
            }
            if (!(placeable ||
playground[i][j].getBackground().equals(shipColor) ||
playground[i][j].getBackground().equals(Color.GRAY))) {
                playground[i][j].setBackground(Color.GREEN);
                playground[i][j].setEnabled(false);
            }
        }
    }
}
```

```java
public interface BattleshipServer extends Remote {
    String method() throws RemoteException;
}
```

```java
public class Server extends UnicastRemoteObject implements
BattleshipServer{

    public Server() throws RemoteException {
        super();
    }
    public static void main(String[] args) throws RemoteException {
        BattleshipServer server = new Server();
        Registry registry = LocateRegistry.createRegistry(1099);
        registry.rebind("BattleshipServer", server);
        System.out.println("Server ready");
    }
    @Override
    public String method() throws RemoteException {
        return null;
    }
}
```

```java
public class Client {
    private final BattleshipServer server;

    public Client() throws RemoteException, NotBoundException {
        Registry registry =
LocateRegistry.getRegistry("26.197.80.44",1099);
        server = (BattleshipServer) registry.lookup("BattleshipServer");
    }
    public String method() throws RemoteException {
        return server.method();
    }
    public static void main(String[] args) throws RemoteException,
NotBoundException {
        Client client = new Client();
        System.out.println(client.method());
    }
}
```