* **Класи во Battleships.Ships :**

1. **Pull Up Method** – Методот **isSunk()** е идентичен кај сите конкретни класи на бродови, па затоа го преместив во апстрактната класа Ship.
2. **Pull Up Method** – Методот **scoreHit()** е идентичен кај сите конкретни класи на бродови освен кај Destroyer, па затоа го преместив и дефинирав во апстрактната класа Ship. Единствено класата Destroyer ќе го препокрие овој метод, бидејќи за неа е различен.
3. **Rename Field** - променливата **remainingIntactCells** во класата Destroyer ја преименував во **intactSegments,** за да соодветствува со променливите од другите конкретни класи за бродови.
4. **Pull Up Field** - Методот **scoreHit()** кoj од конкретните класи го преместив во апстрактната класа Ship, треба да пристапува до променливата intactSegments која се наоѓа во конкретните класи, па затоа оваа променливата **intactSegments** ја преместив во класата Ship (се сетира преку конструктор).
5. **Replace Conditional with Polymorphism -** Методот **shipGridValue(Ship ship**) во класата Ship, проверува од кој тип е бродот зададен како параметар и соодветно враќа некоја вредност. Наместо да имаме услови за проверка на типот на бродот, секоја од конкретните класи за бродови го препокрива методот **shipGridValue()** и секој од нив ја враќа вредноста за тој брод. (Дополнително го отстрануваме параметарот за брод, бидејќи информацијата ја добиваме од бродот кој го повикува методот).
6. **Consolidate Duplicate Conditional Fragments** –во методот placeShipOnGrid **, board.setShipAsPlaced(this)** се повикува и во двете гранки во условот за ориентција на бродот, т.е. и во методот placeVerticalShipOnGrid и во placeHorisontalShipOnGrid. Методот секако ќе се повика, па затоа го преместуваме по if-else делот во методот placeShipOnGrid.

* **Kласа Grid:**

1. Променливите во кои се чуваа бродовите од конкретен тип на брод ги променив да бидат од апстрактен тип Ship за да можам понатаму да ги искористам придобивките на полиморфизмот. За полесно управување со бродовите решив да ги сместам во колекција. Од причина што се потребни методи за конкретните бродови (дали брод од даден тип е поставен или потонат и сл.) избрав бродовите да ги сместам во мапа, така што клучот ќе биде стринг со нивното име. На овој начин лесно го наоѓам бродот од одреден тип и го користам во соодветните методи. Соодветно ги променив методите за додавање и проверка дали е поставен/потонат конкретен брод, да одговараат со претходната промена.
2. **Move field, rename field, pull up field** – boolean полињата **minePlaced, subPlaced, destPlaced …** се преместени во соодветните класи за бродови (move field). Овие променливи имаат иста функција - означуваат дали соодветниот брод е поставен или не, па затоа може да се преименуваат во **isPlaced** (rename field), па бидејќи добиваме иста променлива во сите поткласи, **isPlaced** jа преместуваме во апстрактната класа Ship (pull up field).
3. **Add method** – додадени се следните методи во класата Ship: **setShipAsPlaced()** – ја поставува променливата isPlaced на true, **checkIsShipPlaced()** – ja враќа вредноста на променливата isPlaced.
4. \* - методите за проверка дали некој конкретен брод е поставен (pr. checkMinePlaced()) се сменети согласно претходните две промени – ако тој брод не е креиран се враќа неточно, а ако е, од тој брод се повикува методот checkIsShipPlaced(). Сет методите за поставеноста на бродовите (пр. setMinePlacedTrue ) го повикуваат методот setShipAsPlaced() на конкретниот брод, ако тој е креиран.
5. **Replace conditional with polymorphism** :
   1. Во методот **checkIsShipPlaced(Ship ship)** наместо да проверувам од која класа (односно од кој тип) е дадениот брод, го повикувам методот checkIsShipPlaced() од објектот Ship.
   2. Во методот **setShipAsPlaced(Ship ship)** наместо да проверувам од која класа (односно од кој тип) е дадениот брод, го повикувам методот setShipAsPlaced() од објектот Ship.
   3. Во методот **allShipsSunk()** - бидејќи сите бродови ми се од класа Ship, доволно е да ја изминам мапата во еден циклус и за секој да проверам дали е потонат – не морам да користам методи и проверки за секоја конкретна класа на бродови.
6. **Remove method** - **setAirPlaced(boolean airPlaced)** никаде не се користи со влезен параметар false, па затоа доволно е да го имаме само методот **setAirPlacedTrue().**

Секаде каде што се повикува методот setAirPlaced(true) (во класата GUI), ќе го замениме со повик на методот setAirPlacedTrue().

1. **Rename method** - setAirPlacedTrue() го преименувам во **setAirPlaced(),** за да биде сооодветно името на останатите методи за сетирање на поставеноста на бродовите.
2. **Remove method - isAirPlaced**() и **checkAirPlaced**() ја вршат истата функција па затоа доволен е само еднот од нив. За да биде усогласено името со методите за другите бродови го избрав **checkAirPlaced**(), а другиот метод го избришав. Потоа на секое место каде што се повикува методот isAirPlaced(), го преименував во checkAirPlaced().
3. **Decompose conditional** – во методите **set(int i, int j, int value)** и **update(int i, int j, int value**) ги имаме следните услови:
   1. **(i > userRow || j > userColumn)** = дали координатите се надвор од мрежата.  
      Овој услов го издвојувам во посебен метод **biggerThanGrid (int I,int j)**.
   2. **(i < 0 || j < 0 )** = дали координатите се негативни  
      Овој услов го издвојувам во посебен метод **positionNegative(int I, int j).**
   3. Во методот **isValidPlaceForAShip(int i, int j)** – проверката **(index > 1 && index < 8 )** ја издвојувам во посебен метод **validPlace(int index)**.
4. **Change temp to query** – Во методот **isValidPlaceForAShip(int i, int j)** ја отстранувам привремената променлива **index** и наместо неа директно го користам методот - this.getGridVal(i,j).
5. **Remove method** –Методите **printIsSunk**() и **printIsPlaced**() не се повикуваат никаде па затоа ги избришав.
6. **Rename method - getGridVal(i, j)** го преименував во **getGridValue(int I,int j)** бидејќи името не беше доволно описно.
7. **Remove field**– Во методот **shot(int i, int j),** стрингот **out** никаде не се користи,односно ниту се печати ниту се враќа како резултат, па затоа го избришав.
8. **Consolidate conditional expression** и **Replace conditional with polymorphism**  - во методот **shot(int i, int j)** имаме switch case наредба во која според вредноста на таа позициија се врши различна акција. Ако вредноста на келијата се совпаѓа со вредноста на некој од бродовите, за тој брод се повикува scoreHit(), се ажурира вредноста на таа позиција (се намалува за 8) и се враќа резултат true. Овие услови за проверка ги заменувам така што во еден циклус ги изминувам сите бродови од колекцијата (од абстрактен тип Ship) и ако нивната вредност се совпаѓа со вредноста на таа позиција, ја изведувам акцијата за тој брод (Replace conditional with polymorphism). Остануваат случаите кога вредноста на клетката не се поклопува со вредноста на некој брод. Во сите тие случаи се прави истото – се враќа вредност false, освен во случајот за 0, кога дополнително се прави update на позицијата. Затоа само проверувам дали вредноста е нула за update-от, а не правам проверки за другите случаи, бидејќи во секој случај ќе се врати false (Consolidate conditional expression ).

* **Класа GUI**

1. **Consolidate conditional expression** - Во методите **place[imeNaBrod](int i, int j)** има услови за дали бродот е хоризонтален или вертикален, и во двата случаи се проверува дали додавањето е валидно или не. Ако додавањето е невалидно и во двата случаи (и за вертикален и за хоризонтален брод), акцијата е иста – се печати соодветна порака, а ако додавањето е валидно, тогаш акцијата е скоро иста( со некои мали разлики од 1-2 реда). Затоа, за да ги спојам условите кои враќаат ист резултат, кодот го преуредив, па прво проверувам дали поставувањето е валидно, па потоа дали е бродот вертикален или хоризонтален. На овој начин наместо 2 пати да се повторува кодот при невалидно поставување( и за хор. и за верт.), ќе го има само еднаш (освен во некои случаи кога се разликува, па е направена дополнителна проверка за ориентацијата на бродот).
2. **Replace temp with chain** – во методот **deploy(int i, int j),** ја отстранив привремената променлива **out**, и директно вратив конкатенација од стринговите што ги враќаат повиканите методи.
3. **Decompose conditional** 
   1. условот дали некој брод е на ред за поставување го заменувам со методите **[imeNaBrod]CanBePlaced(int I,int j).**
   2. условот за **endDeploymentPhase**,т.е. проверката дали сите бродови се поставени ја издвоив во посебен метод **allShipsPlaced().**
   3. првиот услов во методот **agentShot(int x,int y**), го издвоив во посебен метод **validForShooting()**
   4. условот за веќе извршено пукање на таа позиција го издвоив во методот **takenShot(int sqrValue).**
4. **Remove method** - Методот **paintComponent** немаше тело и никаде не се повикуваше па го избришав.
5. **Extract method и move method**
   1. Делот за ресетирање на променливите **data.[imeNaBrod]Placed**, односно поставување на нивната вредност на false, го издвоив во посебен метод **resetPlaced().** Бидејќи методот ги користи променливите кои се наоѓаат во класата **BattleShipsEngine**, овој метод го преместив во таа класа.
   2. Делот за ресетирање на променливите **data.agent[imeNaBrod]Sunk**, односно поставување на нивната вредност на false, го издвоив во посебен метод **resetSunk().** Бидејќи методот ги користи променливите кои се наоѓаат во класата **BattleShipsEngine**, овој метод го преместив во таа класа.
   3. ресетирањето на променливите **data.paint[imeNaBrod]Placed**, односно поставување на нивната вредност на false, го издвоив во посебен метод **resetPaintSunk().** Бидејќи методот ги користи променливите кои се наоѓаат во класата **BattleShipsEngine**, овој метод го преместив во таа класа.
   4. делот од методот **reset()** кој се однесува на сетирање на променливи од класата BattleShipsEnginе и не зависи од класата GUI, го издвоив во посебен метод **resetEngine()** и го преместив во класата **BattleShipsEnginе**.
6. **Add methods** – на многу места се повикуваа методи за чие повикување е потребно нижење на објекти, на пр. **data.gameState.playerHomeGrid.check[imeNaBrod]Sunk();** За да се избегне ова направив методи **check[imeNaBrod]Sunk(),** кои во себе го вршат повикувањето на методите од другите класи. Аналогно и за **getGridValue(int I,int j)** кој во себе повикува **data.gameState.playerHomeGrid.getGridVal(i,j);**
7. **Extract class** - Класите ( во истиот фајл со GUI) **AttackMousePressListener** и **HomeMousePressListener** го дефинираат истиот метод **resolveAxisCoOrdinate(int x).** За да не се дуплира кодот, решението е да се креира апстрактна класа која ќе го имлементира методот, а двете класи да ја наследат. Но бидејќи тие веќе наследуваат, а во Јава нема повеќекратно наследување, креирам обична класа AxisResolver со тој метод и чувам референца дo неа во двете класи, преку која класите може да го повикаат методот.
8. **Substitute algorithm** –условите во методот **resolveAxisCoOrdinate(int x)** не беа потребни, бидејќи постои логичка зависност помеѓу х и резултатот, па истите ги заменив со едноставна логика – целобројно делење на х со 20.

* **Класа Agent**

1. **Rename Variable** – Атрибутите кои имаат недоволно описни имиња ги преименував. [**m -> map**] [**g -> grid**]
2. **Remove Method** – Методот **Agent** никаде не се користи затоа го отстранив.
3. **Consolidate Duplicate Conditional Fragments** – Во методот **nextShoot** кога има само еден hotSpot ги споив резултатите од условите во еден. Ова се однесува на поставувањето на позицијата.
4. **Remove Control Flag** – Во методот **nextShoot** ја отстранив привремената променлива **empty** и ја заменив со ***break*** клаузула.
5. **Replace Magic Number with Symbolic Constant** – Наместо да проверуваме дали ќелијата во мрежата е 0, подобро е да провериме дали нема брод на соодветното место.
6. **Replace Temp with Query**
   1. Во методот **nextShoot** ги отстранив привремените променливи **checki, checkj** и **refs.**
   2. Во методот **placeShips** ги отстранив променливите **x, y** и **o.**
7. **Remove Unused Code** – Кодот кој не се користеше го отстранив.
8. **Separate Query from Modifier** 
   1. Методот **placeShips** и менува и враќа резултат. Затоа од него издвоив ***query*** со име **getGrid**.
   2. Методот **nextShot** и ги поставува атрибути и оперира над нив, па затоа ги изолирав, во сетери **setMap** и **setAttackGrid.**
   3. Методите **setSunk** исто така се раздвојуваат од кверијата.
9. **Remove Parametar** – мапата и мрежата кои се проследуваат на **nextShot** веќе не се користат, затоа, ги отстранив.
10. **Hide Method** – Методот **placeShips** го криеме бидејќи веќе не се повикува надвор од самата класа.

* **Класа InfluenceMap**

1. **Remove Variable** – Променливата **maxHotspot** не се користи никаде.
2. **Rename Variable** – Променливите кои имаа несоодветни (недоволно описни) имиња ги преименував: [**maxVal -> maxValue**][**turns -> numberOfTurns**][**hit -> hitValue**] [**coords -> hotspotValue**]
3. **Rename Method** – Методите кои имаа несоодветни (недоволно описни) имиња ги преименував: **getVal -> getValue getMaxHotspotVal -> getHotspotValue**
4. **Replace Magic Number with Symbolic Constant –** Со примената на ова правило ги добивме литералите како **missValue.**
5. **Consolidate Conditional Expression –** Во методот **setDeadends** сите услови имаа иста цел, па затоа ги споив во еден услов.
6. **Decompose Conditional** –
   1. Од условите во методите **getMaxHotspot**, **getNumberOfHotspots**, **getHotspots** и **getIntHotspots** извлековме метод **isHotspot** кој одговара на прашањето дали некоја ќелија е hotSpot.
   2. Од условите во методите **hit, miss** и **sunk** извлеков метод **isHit** и **isMiss** кои одговараат на прашањето дали има погодок или промашување во дадена ќелија.
   3. Од условите во методот **sunk** и **setDeadends** извлеков методи **isLeft**, **isRight isTop**, **isBottom** и **isEdge** кои одговараат на прашањата за позицијата на ќелијата.
   4. Условот во **setDeadends** го издвоив во нов метод **isDeadend**
7. **Extract Method**
   1. Од методите **hit, miss** и **sunk** ги извлеков методите: **miss, hitt, add, substract, compareTo, equals, isOdd, isEven, left, right, above** **below, getWidth** и **getHeight**
   2. Од условите во методот **miss** произлегоа методите **propagateMissLeft, propagateMissRight, propagateMissUp, propagateMissDown.**
8. **Extract Variable –** Од модулите каде се користи вредноста -5 ја извлеков променливата **missValue**
9. **Encapsulate Field –** Полето **coOrds** го енкапсулирав
10. **Extract Class –** Бидејќи постојано се оперира со поединечна вредност од матрицата на вредности издвојуваме класа **InfluenceCell**.
11. **Move Method –** Методите кои оперираат над вредноста на дадена ќелија ги преместив во **InfluenceCell** тоа се: **isHit, isMiss, isHotspot, miss, hit, add, substract, compareTo, equals, isOdd, isEven.**
12. **Move Field –** Атрибитите кои се потребни за методите соодветно ги префрлив во **InfluenceCell.** А тоа се: **missValue, hitValue** и **hotspotValue**
13. **Duplicate Observed Data –** Бидејќи **hotspotValue** е потребно да се користи од **InfluenceCell,** а се пресметува од страна на **InfluenceMap** и зависи од секоја ќелија, ја дуплираме вредноста на полето и обезбедуваме **Observer** шаблон за да се задржи конзистентноста на оваа вредност.
14. **HideMethod –** **get, isLeft, isRight, isTop, isBottom, isEdge, propagateMissLeft, propagateMissRight, propagateMissUp, propagateMissDown, setDeadends, above, below, let, right, computeHotspotValue, isEven, isOdd**
15. **Remove Control Flat –** Во методот **setDeadends** променливаите **done** и **Dave** ги отстранив бидејќи се користат како знаменце.
16. **Replace Temp with Query –** Во методот **getIntHotspots** ја заменив привремената променлива **hsNum** со query.
17. **Replace Exception with Test –** Во методите **hit** и **sunk** ги заменив исклучоците со проверки.
18. **Separate Query from Modifier**
19. **Consolidate Conditional Expression** 
    1. Во методот **setDeadends** сите услови имаа иста цел, па затоа ги споив во еден услов.
    2. Во методот **sunk** исто така се споија условите кои ги делеа ќелиите на централни и на оние кои се на рабови.
    3. Во методот **addMap** споив два услова кои резултираа со ист код.

* **Класа NumberGenerator**

1. **Remove class** - класата не ни помага многу, наместо неа може едноставно да се искористи вградената класа Random од Java

* **Kласа BattleShipsEngine**

1. **Replace magic number with symbolic constant** – вредностите за кои одреден брод треба да биде потонат, ги заменувам со именувана константа, пр. наместо -6, ја користам променливата **mineSunk**.