







actor

1.koordenatuauAukeratu( Koordenatuak pK1, boolean pAurrekoanAsmatu):Koordenatuak

errepikatuta=false

if(!pAurrekoanAsmatu &amp;&amp; this.albokoKoordenatuak.size()==0 )

2.koordenatuRandom(): Koordenatuak

```

k= this.koordenatuRandom();
if(!this.esandakoKoordenatuak.contains(k)){
    errepikatuta=false;
}
else {
    errepikatuta=true;
}

```

do while(errepikatuta)

else if(pAurrekoanAsmatu &amp;&amp; this.albokoKoordenatuak.size()!=0)

3.albokoKoordenatuakSortu(pK1):void

kont=0

while(kont &lt; 4)

4.this.zentzuBateanKoordenatuBerriak(auxPX, auxPY, kont):Koordenatuak

5.zeinErtzaDa(px, py):short

```

ertzZenb = 0;
if(this.zeinIzkinaDa(px, py) == 0)

```

6.zeinIzkinaDa(px, py):short

izkinaZenb=0

7.izkina1Da(px, py):boolean

if(this.izkina1Da(px, py))

izkinaZenb= 1;

8.izkina2Da(px, py):boolean

else if(this.izkina2Da(px, py))

izkinaZenb= 2;

9.izkina3Da(px, py):boolean

else if(this.izkina3Da(px, py))

izkinaZenb= 3;

10.izkina4Da(px, py):boolean

else if(this.izkina4Da(px, py))

izkinaZenb= 4;

11.ertz1Da(px, py): boolean

if(this.ertz1Da(px, py))

ertzZenb=1;

12.ertz2Da(px, py): boolean

if(this.ertz2Da(px, py))

ertzZenb=2;

13.ertz3Da(px, py): boolean

if(this.ertz3Da(px, py))

ertzZenb=3;

14.ertz4Da(px, py): boolean

if(this.ertz4Da(px, py))

ertzZenb=4;

15.geituAlbokoKoordenatuak(this.zentzuBateanKoordenatuBerriak(auxPX, auxPY, kont):void

kont++

```

int konti=0;
do {
    k=this.albokoKoordenatuak.get(this.zentzia);
    konti++;
    if (this.zentzia==3) {
        this.zentzia=0;
    }
    else {
        this.zentzia++;
    }
}while(k.getKoordenatuakY() == -1 && k.getKoordenatuakX() == -1 && this.esandakoKoordenatuak.contains(k) && konti<=4);
if(konti==4) {
    errepikatuta=false
}

```

16.koordenatuRandom():Koordenatuak

```

k= this.koordenatuRandom();
if(!this.esandakoKoordenatuak.contains(k)){
    errepikatuta=false;
}
else {
    errepikatuta=true;
}

```

do while(errepikatuta)

else if(pAurrekoanAsmatu &amp;&amp; this.albokoKoordenatuak.size()!=0)

int aux=0;

int zentzia= this.zentzia;

do {

aux++;

k=this.albokoKoordenatuak.get(this.zentzia);

if (this.zentzia==3) {

this.zentzia=0;

}

else {

this.zentzia++;

}

}while(k.getKoordenatuakY() == -1 &amp;&amp; k.getKoordenatuakX() == -1 &amp;&amp; this.esandakoKoordenatuak.contains(k) &amp;&amp; aux&lt; 4-zentzia);

if (aux== 4-zentzia) {

errepikatuta= false;

22.koordenatuRandom():Koordenatuak

```

k= this.koordenatuRandom();
if(!this.esandakoKoordenatuak.contains(k)){
    errepikatuta=false;
}
else {
    errepikatuta=true;
}

```

do while(errepikatuta)

if (this.zentzia==0)

23.koordenatuRandom()

```

k= this.koordenatuRandom();
if(!this.esandakoKoordenatuak.contains(k)){
    errepikatuta=false;
}
else {
    errepikatuta=true;
}

```

do while(errepikatuta)

24.erreseteatuAlbokoKoordenatuak()

# Proba kasuen deskribapena

OHARRA: Metodo pribatu batzuen proba kasuak deskribatu ditugu, errazago egiteko metodo horiek erabiltzen dituzten metodo publikoen proba kasuak.

## ● Koordenatuak:

### -Eraikitzailea:

Bi eraikitzaile daude, bata parametrorik gabe eta bestea 2 parametroekin.  
new Koordenatuak() egiten denean, konprobatu behar da x eta y atributuak -1 balioa hartzen dutela, hori baita eraikitzailean zehaztuta dagoena.  
new Koordenatuak(pX, pY) egiten denean, konprobatu behar da x eta y atributuak pX eta pY balioak hartzen dutela, hurrenez hurren.

### -setKoordenatuakX(pX):

X atributuaren balioa aldatzen du eta bere balio berria pX dela konprobatu behar dugu.

### -setKoordenatuakY(pY):

Y atributuaren balioa aldatzen du eta bere balio berria pY dela konprobatu behar dugu.

### -getKoordenatuakX():

X atributuaren balioa bueltatzen duela ziurtatu behar dugu.

### -getKoordenatuakY():

Y atributuaren balioa bueltatzen duela ziurtatu behar dugu.

## ● JokalariCPU:

### -koordenatuaAukeratu(pK1, pAurrekoanAsmatu):

Aurrekoan itsasontzik ez badu ukitu, guztiz ausazko koordenatu bat eratuko duela konprobatu behar dugu. Aurreko txandan itsasontzia ukitu badu, koordenatu berria aurretik esandako koordenatuaren alboan esango duela konprobatu behar dugu. Itsasontzi baten bi kasila ukitu baditu, badaki zein norabidean dagoen (horizontalean edo bertikalean), beraz, bakarrik noranzko horretan bilatuko du eta, zentzu batean bilatuko du noranzko horretan. Ura aurkitzen badu zentzu horretan, noranzko berdinean bilatuko du baina kontrako zentzuan; hau da, horizontalean dagoen itsasontzi bat aurkitu badu, eskuinerantz bilatzen jarraituko du, baina ura aurkitzen badu eta orain arte aurkitutako itsasontziaren luzera aurkitu ditzakeen itsasontzikirik luzeenaren luzera baino txikiagoa da, ezkerrerantz bilatuko du.

Hau da metodo nagusiena eta publikoa da. Beraz hurrengo metodo pribatuak ondo dauden konprobatzeko modu bakarra metodo hau erabiliz da. Eta ondorioz honekin baita probatu behar ditugun koordenatuak honako proba kasu hauek ondo dauden ikusteko erabiliko da:

**-izkina1Da(pX, pY):**

Sartutako koordenatuak tableroaren goiko eskumako izkinakoak badira true bueltatuko duela konprobatu behar dugu.

**-izkina2Da(pX, pY):**

Sartutako koordenatuak tableroaren goiko ezkerreko izkinakoak badira true bueltatuko duela konprobatu behar dugu.

**-izkina3Da(pX, pY):**

Sartutako koordenatuak tableroaren beheko ezkerreko izkinakoak badira true bueltatuko duela konprobatu behar dugu.

**-izkina4Da(pX, pY):**

Sartutako koordenatuak tableroaren beheko eskumako izkinakoak badira true bueltatuko duela konprobatu behar dugu.

**-zeinIzkinaDa(pX, pY):**

Sartutako koordenatuak tableroaren izkina bat badira, zenbaki bat batetik laura bueltatuko duela konprobatu behar dugu. Goiko eskumako izkina bada, 1 bueltatuko du; goiko ezkerreko izkina bada, 2 bueltatuko du; beheko ezkerreko izkina bada, 3 bueltatuko du eta 4 bueltatuko du beheko eskumako izkina bada.

**-ertz1Da(pX, pY):**

Tableroaren eskumako ertzan badaude sartutako koordenatuak, true bueltatzen duela konprobatu behar dugu.

**-ertz2Da(pX, pY):**

Tableroaren goiko ertzan badaude sartutako koordenatuak, true bueltatuko duela konprobatu behar dugu.

**-ertz3Da(pX, pY):**

Tableroaren ezkerreko ertzan badaude sartutako koordenatuak, true bueltatuko duela konprobatu behar dugu.

**-ertz4Da(pX, pY):**

Tableroaren beheko ertzan badaude sartutako koordenatuak, true bueltatuko duela konprobatu behar dugu.

**-zeinErtzaDa(pX, pY):**

Sartutako koordenatuak tableroaren ertz batean badaude, batetik laurako zenbaki bat bueltatuko du. Eskumako ertzan badaude, 1 bueltatuko duela konprobatu behar dugu, 2 goiko ertzan badaude, 3 ezkerreko ertzan badaude eta 4 beheko ertzan badaude.

**-kontrakoZentza():**

“Zentza” JokalariCPU-ren atributua aldatzen du. 0 balioa badauka, 2ra aldatuko duela konprobatu behar dugu. 1ean badago, 3ra aldatuko duela konprobatu behar dugu. 2 balioa badauka, 0ra aldatuko duela konprobatu behar dugu eta 3 balioa badauka, 1 baliora aldatuko duela konprobatu behar dugu.

**-zentzuBateanKoordenatuBerriak(pX, pY, pZentza):**

Koordenatu bat emanda, zentzuaren arabera alboko koordenatu bat bueltatuko du, 0 bada, eskumakoa; 1 bada, goikoa; 2 bada, ezkerrekoa eta 3 bada, behekoa. Hori konprobatzearaz gain, koordenatu hori eratu ahal ez izatekotan, -1, -1 koordenatuak bueltatzen duela konprobatu behar dugu. Adibidez, goiko eskumako izkina sartzen badiogu eta 0 zentza, ezin da koordenatu hori eratu, beraz, -1, -1 balioa bueltatuko du, eta horrela ezinezko koordenatu guztiekin.

**-erreseteatuAlbokoKoordenatuak():**

albokoKoordenatuak atributua erreseteatzen duela eta barruan dituen elementu guztiak ezabatzen dituela konprobatu behar dugu.

**-gehituAlbokoKoordenatuak(pK):**

albokoKoordenatuak atributuan pK koordenatuak gehitzen dituela ziurtatu behar dugu.

**-albokoKoordenatuakSortu(pK):**

pK koordenatuak hartuz, lista bat egiten du non 4 balio daude: lehenengo posizioan pK-ren eskumako koordenatuak, bigarren posizioan goiko koordenatuak, hirugarren posizioan ezkerreko koordenatuak eta laugarren posizioan beheko koordenatuak. Horrez gain, konprobatu behar dugu -1, -1 balioa gehitzen duela listan koordenatu hori eratu ezin bada.

**-koordenatuRandom():**

Tableroan sartzen den ausazko koordenatu bat bueltatzen duela konprobatu behar dugu.

**-itsasontziakJarri(int pErrenkadaZutKop):**

Metodo hau randomly egiten du guztia, eta gainera tableroan dagoen metodo bati dei egiten dio. Tableroko metodo hau TableroTest-an konprobatzen da ondo dabilen ala ez. Beraz, hainbat aldiz deituko diogu metodo honi eta inprimatuko dugu tableroa dei bakoitzean eta ikusiko dugu itsasontziak ondo daude jarrita.

- **Tableroa:**

Klase honen proba kasuak egiteko erabiliko dugu *KoordenatuanJarri* (*int pX, int pY, String pJarri*) metodoa. Metodo honek ahalbidetzen digu edozein koordenatuan nahi duguna sartzen. Horrela koordenatu batean “ U” sartu ahal dugu eta ikusi ea metodoak itzultzen duen itzuli behar duena.

**-getErrenkadaZutKop():**

Tableroaren errenkada/zutabe kopurua bueltatzen duela konprobatu behar dugu.

**-eguneratuTableroa(pX, pY, pEmaitza):**

Etsaiak bere tableroan koordenatu horietan zuena bere tableroan jartzen du. Ura bazeukan X jarriko duela eta itsasontzia ukitu badu U jarriko duela konprobatu behar dugu. *KoordenatuanZerDagoen(short pX, short pY)* metodoa erabiliko dugu.

**-tableroaBete():**

Bakarrik inprimatuko dugu eta ikusiko dugu uraz (“ - ”) beteta dagoela.

**-tableroalInprimatu():**

Bakarrik inprimatuko dugu eta ikusiko dugu tableroa inprimatzen duela errenkada eta zutabeen zenbakiekin.

**-itsasontziakJarri(short pX, short pY, short pltsas, String pOrientazio):**

Hurrengo metodoen kasuak aztertu behar dira:

**-konprobatuHutsuneak(pX, pY, pltsas, pOrientazio):**

Orientazioa horizontala bada (H,h) konprobatu behar dugu itsasontzi bat ez egotea bere inguruan. Itsasontzia edozein lekuaren jarriz gero, baita izkinak eta ertzak kontutan izanda, bere inguruan itsasontzi bat ez badago konprobatu behar dugu true bueltatzea. Eta itsasontzi bat badago konprobatu behar dugu false bueltatzea. Eta gauza bera orientazioa bertikala (B,b) denean.

**-konprobatultsasontzirikEzKoordenatuan(short pX, short pY, short pltsas, String pOrientazio)**

Orientazioaren arabera konprobatu behar dugu sartu nahi dugun koordenatuan itsasontzi bat badago false itzultzen duela eta bestela true itzultzen duela. Baita gerta daiteke sartu duzun koordenatuan itsasontzirik ez egotea baina jarritako orientazioaren eta itsasontziaren tamainaren arabera itsasontzi baten gainean jarri nahi izatea. Orduan konprobatu behar dugu horrelako kasuetan false bueltatzen duela eta bestela true.

**-konprobatuTiroa (short pX, short pY):**

Konprobatu behar dugu koordenatu horretan ura ez badago, hau da, desberdin “ - ” bada false itzultzen duela. Eta koordenatu horretan ura badago, hau da, berdin “ - ” bada true itzultzen duela. Baita konprobatu behar dugu koordenatu horretan itsasontzi bat ukitu baduzu, hau da, berdin “ U” bada false itzultzen duela eta bestela true. *KoordenatuanJarri* (*int pX, int pY, String pJarri*) metodoa erabiliko dugu.

#### **-koordenatuanZerDagoen(short pX, short pY):**

Konprobatu behar dugu koordenatu horretan itsasontzi bat badago, hau da, “ 1”, “ 2”, “ 3” edo “ 4” badago “ U” itzultzen duela eta bestela koordenatu horretan dagoena itzultzen duela, hau da, “ -” itzultzen duela. *KoordenatuanJarri ( int pX, int pY, String pJarri)* metodoa erabiliko dugu.

#### **-KoordenatuanJarri ( int pX, int pY, String pJarri):**

Konprobatu behar dugu koordenatuan “ U”, “ 1”, “ 2”, “ 3”, “ 4”... sartzen badugu koordenatu horretan String hori dagoela. Horretarako *assertEquals* eta *koordenatuanZerDagoen(short pX, short pY)* metodoa erabiliko dugu.

- **BikoteJokalariak:**

#### **-partidaBatJokatu()**

##### **-itsasontziakJarri()**

Konprobatu behar lehenik tableroa betetzen duela uraz( -) eta gero itsasontziak jarri dituela, eta hori konprobatzeko ez dagoenez assert bat, kontsolatik konprobatzen dugu.

##### **-partidaBukatu**

Jokalari klaseak duen eguneratuPrintTablero() metodoa erabiliko dugu nUkituta guk nahi dugun balioa izan dezan partida noiz bukatzen den guk erabakitzeko eta ondo funtzionatezen duenentz jakiteko. Edozein jokalariko nUkitua==10 denean partidaBukatu true bueltatuko du eta partida amaituko da. nUkitua=10 jartzeko *setNUkitua* metodoa erabiliko dugu.

##### **-koordenatuaAukeratu()**

Honako metodo hau kontsolatik konprobatu behar dugu JokalariArruntatik (JokalariCPU-k dena random egiten duelako), kasu bakoitza gertatzeko koordenatuak sartzen:

- lehen esan ez dugun koordenatua
- errepikatu koordenatua
- sartutako koordenatuarekin itsasontzirik ez ukitu
- sartutako koordenatuarekin itsasontzia ukitu

#### **-getNireBikoteJokalariak()**

BikoteJokalariak klaseko bi erakusle sortuko ditugu (*BikoteJokalariak.getNireBikoteJokalariak()* ) eta *assertEquals* batekin berdinak direla konprobatuko dugu, BikoteJokalariak EMA bat baita.

- **Jokalaria:**

#### **-nireTableroAlete()**

Konprobatu beharko dugu Tablero klasean TableroAlete() metodoak, nireTablero attributu moduan dugun Tablero uraz (“ -” ) betetzen duela, hori konprobatu ahal izateko ez dagoenez inolako assert bat , inprimatuko dugu eta kontsolatik konprobatuko dugu.

#### **-getNireTablero()**

Egiaztatu beharko dugu Tablero bat bueltatzen digula, hain zuzen ere, *NireTablero*.

#### **-getPrintTablero()**

Frogatu beharko dugu Tablero bat itzultzen digula, baina kasu honetan *PrintTablero*.

#### **-itsasontzirikEz()**

Metodo honek boolean bat bueltatuko du, kasu honetan true, itsasontzirik ez badago tableroan, hau da, *nUkituta*=10 bada. Beste kasu guztietañ false bueltatu beharko luke.

#### **- koordenatuanZerDagoen(short pX, short pY)**

Metodo honek daukan zeregina da jasotzen dituen koordenatuetan zer dagoen *nireTablero-an* itzultzea da (String bat), horretarako egiten duena da *Tablero* klasean dagoen *koordenatuanZerDagoen(pX,pY)* deitzea da, konprobatu beharko duguna Tablero klasekoa izango ohi da, baita espero ditugun String-nak jasotzen ditugula.

#### **-kordenadaBaliogarriak(short pX, short pY)**

Bi koordenada jasota, *nireTablero* Tablero klasean dagoen *konprobatuTiroa(pX,pY)-ari* deitu egingo dio eta honen menpean geratuko da kordenadaBaliogarriak bueltatuko duen erantzuna True edo False.

#### **-eguneratuPrintTablero(short pX, short pY, String pEma)**

Jasotzen dugun *pEma* izango da, *koordenatuanZerDagoen(pX,pY)* metodoan jasotzen dugun String-na. Orduan begiratzeko beharra dago, jasotzen dugun String-na “U” bat bada edo ez, “U” bat izanda, lehenengo gauza egin beharrekoa da, *nUkituta* inkrementatzea horretarako metodo bat badaukagu. Baina “U” bat izanda ala ez, Tablero klasean dagoen *eguneratuTablero(pX,pY,pEma)*-ri deitzeko beharra dago, kasu honetan *printTablero*-rekin egingo da deia, eta *pEma* matrize horren *pX* eta *pY* koordenatuetan jarriko da.

#### **-nUkitualInkrementatu()**

Metodo honen helburua *nUkituta* duen balioa inkrementatzean datza. Frogatzeko bere helburua ondo egiten duela, nahikoa izango da, bi balio assert batekin konparatuz, bata espero duguna, eta bestea metodoak sortuko duena. Horretarako *getNUkitualInkrementatu()* metodoa erabiliko dugu.

#### **-getIzena():**

Konprobatu beharko dugu, izena atributuaren balioa bueltatzen duela.

#### **-setNUkituta(int pUkituta):**

Konprobatu behar dugu balioa aldatu duela. Horretarako assert bat egingo dugu, espero dugun eta duena konparatuz. *getNUkituta()* erabiliko dugu horretarako.

**-getNUkituta():**

Konprobatu beharko dugu, nUkituta atributuaren balioa bueltatzen duela.

- **JokalariArrunta:**

**-koordenatuaAukeratu():**

Konsolatik sartzen ditugun zenbakiak koordenatu bezala bueltatzen direla ziurtatu behar dugu. Sartuko dugun lehenengo zenbakia zutabea eta bigarren zenbakia errenkada izango dela konprobatu behar dugu ere bai.

**-itsasontziakJarri(int pErrenkadaZutKop):**

Sartuko den lehenengo zenbakia zutabea izango dela eta bigarrena errenkada ziurtatu behar dugu. Horrez gain, itsasontziak ordenean jarriko direla konprobatu, hau da, lehenengo itsasontzia kasila batekoa izango dela, bigarrena bi kasilakoa, hirugarrena hirukoa eta azkena laukoa. Salbuespenak daudenez metodo honetan, horiek bebai konprobatu behar ditugu. Tableroan ez dagoen koordenatu bat sartzen denean, salbuespen egokia saltatuko duela konprobatu behar dugu, gainera, tableroan dagoen koordenatu bat esaten badugu, baina sartu behar dugun itsasontzia ez bada sartzen, mezu egokia inprimatu behar da. Azkenik, tableroa modu egokian inprimatzeko delako itsasontzi bat jartzean konprobatu behar dugu.