**< Програм хангамжийн хөгжүүлэлт >**

(Дадлага ажил 2)

А. Дэмбэрэлдодов

ХШУИС, Програм хангамжийн 3 курсын оюутан, [20B1NUM0369@stud.num.edu.mn](mailto:20B1NUM0369@stud.num.edu.mn)

# Оршил

Энэхүү дадлага ажил 2 хүрээнд жава хэлтэй танилцах болон argoUML дээр классын диаграм код руу хөрвүүлэх.

# Зорилго

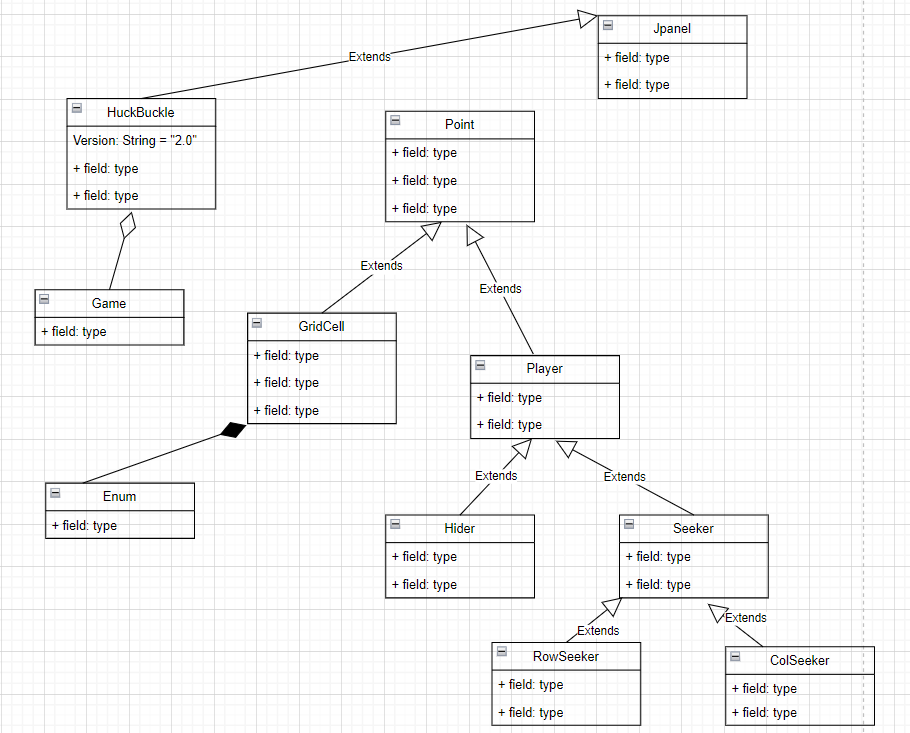
Лекцүүдийн хүрээнд үзэх шаардлагатай агуулгуудыг судлаж, өөрийн үгээр тайлбарлаж тэмдэглэл бичих. Номны дасгалыг уншиж, ойлгож ажиллах. Зорилгодоо хүрэхийн тулд дараах зорилтуудыг дэвшүүлсэн байгаа.

1. Лекцийн материал болон лекцийн үзэж ойлгох

Дасгал1:

**Бүлэг даалгавар 1 : Кодыг нягтлан хянах, урвуу инженерчлэл**

a. Классууд нь хэрхэн уяалдаж буйг судлахын тулд hbbv2.0 –ийн эх кодыг нягтлан хянаж уншина уу. Энэ зураг дах класс хоорондын удамшил (inheritances), хэрэгжүүлэлт (realizations), холбоос (associations) зэргийг бүгдийг сайн харагдахуйц болгож Зураг 1 дэх класс диаграмийг өөрчилнө үү. Бүх холбоосын жолоодож болохуйц байдал (navigability) болон харьцааны тоо (multiplicity)-г харуулах ёстой. Диаграмд ядаж нэг нарийвчилсан холбоосыг (ж. нь бүрдэл (aggregation) эсвэл нийлмэл (composition)) харуулах ёстой бөгөөд энэхүү холбоосын нарийвчилсан шийдлийг дэд асуулт в-д баримтжуулж өгнө. Өөрчилсөн диаграмд ямар нэгэн инстанцийн хувьсагч (instance variables) эсвэл арга (methods)-ыг харуулах шаардлаггүй.



b. Класс диаграм дах холбоосуудаас нэгийг нь яагаад бүрдэл эсвэл нийлмэл болгож нарийвчилсан талаар дэлгэрэнгүй тайлбарлана уу. Оноо авахын тулд “агуулах”, “-тай байх”, “өөрийн болгох” (“own”) болон устгах (“destroy”) гэсэн үгсийг ойлгомжтой, зохистой ашиглана, мөн ямар холбоосын тухай ярьж байгааг илэрхий цохон заахаас гадна яагаад энэ холбоосыг сонгосон тухай ойлгомжтой байх ёстой.

HuckleBuckle классын Game класстай бол бүрдмэл харьцаагаар холбогдсон байгаа.

*myGame* = new Game(gridSize);

HuckleBuckle устгах үед Game класс бүрэн устгахгүй. Учраас бүрдмэл харьцаа байгаа юм.

**Бүлэг даалгавар 2 : Set interface-ийн тухай сурах**

(5 оноо) “Java Tutorials” –ийн The Set Interface, JavaDoc definition of the Set interface болон add() method гэсэн хэсгүүдийг уншина уу. Унших зорилго бол Set интерфейсийн агуулга зүйн гол санаа болон код бичихдээ хэрхэн хэрэглэх санааг ойлгох юм. Энд “contract” гэсэн ойголтын тухай, Set интерфейсийн төлөв байдлыг (үйлдлийн иж бүрдэл) тодорхойлох гэсэн формал утгатай, илүү ихийг мэдэхийг хүсвэл Design by contract –ийг уншаарай. Гүйцэтгэх ажил: HuckleBuckle() байгуулагч болон Player классын “TODO comment” –ийг уншина уу. Эдгээр нь hbbv2.0 –ийн төлөв байдлын согогийн талаар өгүүлсэн. Player классыг сайжруулсан hbbv2.1–ийг үүсгэээрэй, ингэснээр HuckleBuckle-ийг олон Seeker тоглох боломжтой болно. Оноог бүрэн авахын тулд өөрийн бичсэн кодны бүх аргуудын тайлбар нь товч тодорхой, мэдээлэл өгөхүйцээр байх хэрэгтэй. Мөн кодны давхардалгүй байх хэрэгтэй

1. package hucklebuckle;
2. import java.awt.\*;
3. abstract class Player extends Point {
4. // constraint: 0 <= x < myGame.gridSize()
5. // constraint: 0 <= y < myGame.gridSize()
6. private static final long serialVersionUID = -5253550170404888625L;
7. private String myName;
8. private Game myGame; // note: uninitialised for new Players
9. //TODO: override the equals() method of Point, so that a Set<Seeker>
10. //can have multiple Seekers at the same location of the grid.
11. @Override
12. public boolean equals(Object obj) {
13. return this == obj;
14. }
15. Player(String myName) {
16. this.myName = myName;
17. setLocation(0,0); // all players start at (0,0)
18. }
19. String getName() {
20. return myName;
21. }
22. Game getGame() {
23. return myGame;
24. }
25. void setGame(Game myGame) {
26. this.myGame = myGame;
27. }
28. void setX(int px) {
29. setLocation(px, getY());
30. }
31. void setY(int py) {
32. setLocation(getX(), py);
33. }
34. @Override
35. public String toString() {
36. return getName() + " at (" + (int) getX() + ", " + (int) getY() + ")";
37. }
38. }
39. HuckleBuckle class:
40. myGame.equals(allSeekers);
41. (6 оноо) Энд hbbv2.1 –ийн hbbv2.1а гэсэн “турших” хувилбарыг үүсгэх ба үүгээр HuckleBuckle классын байгуулагч доторх allSeekers гэх хувьсагчийн олон янзын хэрэгжүүлэлт болон төрлийг туршиж үзэх болно. Энэ бол дотоод хувьсагч бөгөөд иймэрхүү хувьсагчийн агуулга зүйн үндсэн санааг лекцийн материалаас харна уу. Энэхүү хувьсагчийг Game классын setSeekers() –ийг дуудахад, мөн Hider классын pleasePlayWith() дуудахад гэсэн хоёр янзаар хэрэглэж байна. Цуглуулгын фреймворк дах Set-ийн TreeSet болон LinkedHashSet гэсэн өөр хоёр хэрэгжүүлэлтийг HuckleBuckle –ийн байгуулагчийн гурав дах мөр дээр дуудсан хэсгийг өөрчилж туршиж үзээрэй. Энэ өөрчлөлт нь ямар нэгэн орчуулгын алдаа, ажиллах үеийн алдаа, эсвэл ажиглагдахуйц төлөв байдлын өөрчлөлт гэх мэтийн учир шалтгаан болж байна уу? allSeekers -ийн өөр бусад төрлийг туршиж үзнэ үү: Set -ийн оронд Collection гэж зарлах, List болон List –аар зарлах. Зохих импортыг хийсний дараа, энэ гурван тохиолдолд ямар нэгэн орчуулгын алдаа гарч байна уу? Хэрэв тийм бол энгийн өөрчлөлт хийж согогийг хялбар засварлаж байна уу эсвэл өөр классыг өөрчлөх хэрэгтэй юу? Өөр классыг өөрчилсөн бол энэхүү өөрчлөлт нь нийт кодыг сайжруулж байна уу эсвэл HuckleBuckle классын шинэ хэрэгжүүлэлтэд тохируулахын тулд олон класст өөрчлөлт хийж байна уу (өөр зарим нэгэн хэрэгжүүлэлтэд тохируулахын тулд, ирээдүйд зарим нэгэн хөгжүүлэгчид “түүнийг дахин өөрчлөх эрсдэл” )? Энэ туршилтын гурван хувилбарыг үүсгэж ажиллуулсаны дараа allSeekers заалтан хувьсагчийн төрлийг өөрчлөснөөс шалтгаалсан ажиллагааны байдлын ямар нэгэн ажиглагдахуйц өөрчлөлт үүссэн эсэхийг олж тогтооно уу. Одоо “Java Tutorials” –ийн Set Interface дэх “Note that the code always refers to the Collection by its interface type…” гэсэн цогцолборыг нягтлан хянаж анхааралтай уншина уу. Дээрх туршилтыг хийхэд гарсан асуудал, үр дүн, ач холбогдлын талаар нарийн дэлгэрэнгүй тайлан бичиж илгээнэ үү. hbbv2-д JCF-ийг ашигласан өөрчлөлтийн аргачлал, кодод хийсэн өөрчлөлт хялбар, хэцүү байсан эсэх, учир шалтгааныг тайлбарлана.

AllSeekers-ийн TreeSet хэрэгжилт рүү шилжсэнээр ажиллах хугацаа үүссэн.

Үл хамаарах зүйл, учир нь Seeker Comparable интерфэйсийг хэрэгжүүлдэггүй ч TreeSet<S> нь compare()-г ашигладаг. Энэ тоглоомын оролцогч талууд үүнийг тохиромжтой гэж үзэж магадгүй юм хайгчдыг нэрээр нь цагаан толгойн дарааллаар, эсвэл тоглоомд нэгдсэн дарааллаар нь эрэмбэлэж байна. Эдгээр захиалгын аль нэг нь тоглоомын семантикт мэдэгдэхүйц өөрчлөлт оруулах тул TreeSet руу шилжих боломжгүй юм. . AllSeekers-г LinkedHashSet хэрэгжүүлэлт рүү шилжүүлснээр эмхэтгэл гарсангүй алдаа (би холбогдох багцыг импортолсоны дараа). Би ажлын цагийн үр ашгийн хувьд ямар ч ялгаа олж хараагүй тул би үүнийг бодохгүй байна

өөрчлөлт нь сайжруулалт болно. AllSeekers-ийн төрлийг Цуглуулга<Seeker> руу шилжүүлснээр эмхэтгэл бий болсон

амархан зассан алдаа: Би Тоглоомын mySeekers жишээ хувьсагчийн төрлийг ерөнхийд нь дүгнэсэн.

Set<Seeker> гэхээсээ илүү Collection<Seeker> ба (мэдээж) би хүлээн авагч болон тохируулагчийн гарын үсгийг ерөнхийд нь бичсэн.

Тэд тусгай төрлийн цуглуулга шаарддаггүй. Би энэ өөрчлөлтийг бага зэрэг сайжруулсан гэж хэлмээр байна

**Бүлэг даалгавар 3 : Төлвийн диаграмийг турших – Танилцуулга**

1. (6 оноо) hbbv2.0 –ийн Seeker классын эх кодыг судална уу. Мөн Зураг 2 –т үзүүлсэн төлвийн диаграмийг харна уу. Төлвийн диаграм нь Seeker –ийн амьдралын мөчлөгийг ойлгоход тустай бөгөөд програм хөгжүүлэхэд өргөн хэрэглэдэг техникийн нэг юм. Бодит байдалтай адилгүй, Seeker нь үүссэний дараа тэр даруй WAITING-ХҮЛЭЭЖ БУЙ төлвөөс эхэлдэг, дараа нь (тодорхой аргын дуудалтыг хүлээж авсаны дараа) STARTING-ЭХЛЭЖ БУЙ эсвэл MOVING-НҮҮЖ БУЙ гэх мэтийн бусад төлөв рүү шилжинэ. Төлөвийн диаграмийн ойлголтоо баталгаажуулахын тулд Seeker -ийн move() аргын эхний мөр дээр зогсох цэг (breakpoint) -ийг тохируулж өгөөрэй. Зогсох цэг дээр очих бүрт Seeker – ийн гишүүн myState –ийг шалгаж үзээрэй. Eclipse-ээр “debug-алдаа илрүүлж засварлах” хийх талаар бие даан судлаарай.

Гүйцэтгэх ажил: Шинэ төсөл (project) hbbv2.2 –ийг үүсгээд, өөрийн hbbv2.1 –ийн эх кодыг импорт хийнэ. Шинэ төлөв ASKING –ийг SeekerState enum руу нэмж өгнө үү. Одоо зорьж буй байрлал (nextX, nextY) –т очихын тулд хангалттай алхам хийсний дараа Seeker нь ASKING (SEEKING төлвийн оронд) төлөв рүү орсон байхаар тохируулж Seeker -ийн move()аргын хэрэгжүүлэлтийг өөрчилнө үү. ASKING төлөвт Seeker нь Hider –ээс яг одоогийн температурыг асуух ёстой. Температур нь FOUNDIT биш бол SEEKING төлөвт цааш үргэлжлэх ёстой. Боломжит бүх нуусан газарт хайж дууссан эсэхээс хамааран тоглоомыг дуусгахаар шийдэх юм уу эсвэл (nextX, nextY) –ийн шинэ утгыг олж тогтоох хоёрын аль нэгийг SEEKING төлөвт гүйцэтгэнэ. Энэхүү өөрчлөлтийн үр нөлөө нь хайгчийн ахицыг удаашруулах гэдгийг анхаарах хэрэгтэй, учир нь дараагийнх руу нүүхээс өмнө нүд бүр дээр цагийн хоёр агшинг (timer tick) зарцуулна. Хэрэв температур нь мэдэгдэж байвал, энэхүү өөрчлөлт нь шинэ төрлийн хайгчийн хувьд нүднүүдээр хурдтай нүүх боломжийг бас олгоно (ж.нь нуугчаас температурыг асуухгүйгээр (ASKING)).

public boolean move() {

switch (myState) {

 case WAITING:

break; 7

 case STARTING: nextX = 0;

first nextY = 0;

State = MOVING;

 break;

 case MOVING:

 if ( (getX() == nextX) && (getY() == nextY ) )

{

// I have reached my destination

myState = ASKING; }

 else

 { // I move one step toward (nextX, nextY) translate((int)

 Math.signum(nextX - getX()), (int) Math.signum(nextY - getY()) ); distanceMoved++;

 // Note: moveTo() is no longer called in tryNewPosition() System.out.println(getName() + " says, \"I took a step. I'm now at " + (int) getX() + ", " + (int) getY() + ".\""); myState = MOVING; // I'm still moving } break; case ASKING: System.out.println(getName() + " asks, \"Am I close?\""); if (myHider.pleaseRevealTemperatureOf(this) == Temperature.FOUNDIT) { myState = WINNING; // Hooray! } else { myState = SEEKING; } break; case SEEKING: if (tryNewPosition()) { // This method updates (nextX, nextY) myState = MOVING; // I keep looking } else { myState = QUITTING; // I give up! } break; case QUITTING: // I don't move, but I do say something before WAITING. System.out.println(getName() + " says, \"I give up! I took " + distanceMoved + " steps before quitting.\""); myState = WAITING; break; case WINNING: // I don't move, but I do say something before WAITING. System.out.println(getName() + " says, \"That was fun! I walked " + distanceMoved + " step" + (distanceMoved != 1 ? "s" : "") + " before I found it.\""); myState = WAITING; break; } return (myState != WAITING); }

Playing HuckleBuckle (v2.1) on a 5 by 5 grid... Harry says "Hi Sue, I'm Harry, let's play Huckle Buckle!" Sue says, "Hi, Harry, I'm Sue. Glad to meet you! Are you ready?" Harry says "Hi Sally, I'm Harry, let's play Huckle Buckle!" Sally says, "Hi, Harry, I'm Sally. Glad to meet you! Are you ready?" Harry says to everyone, "I'm ready now. I bet you can't find it!" Sue asks, "Am I close?" Harry says to Sue, "You're cold." Sally asks, "Am I close?" Harry says to Sally, "You're cold." Sue says, "I took a step. I'm now at 0, 1." Sally says, "I took a step. I'm now at 1, 0." 8 Sue asks, "Am I close?" Harry says to Sue, "You're cool." Sally asks, "Am I close?" Harry says to Sally, "You're cool." Sue says, "I took a step. I'm now at 0, 2." Sally says, "I took a step. I'm now at

**Бүлэг даалгавар 4 : Гурван товчлуур нэмэх 5. (6 оноо) Java Tutorials –ийн хичээл**

Using Swing Components-ийн эхний хэдэн хуудсыг уншаад, Using Text Components, How to Make Applets гэсэн хуудсыг алгасаад, How to Use Buttons, Check Boxes, and Radio Buttons гэсэн хуудсыг дуустал уншина. Top-level Demo -ийг уншсан хэсэгт санал санал болгосны дагуу импортлож, орчуулаад, ажиллуулна. Swing app–ийн иймэрхүү нэгэн энгийн үлгэр загвар код нарийн ээдрээтэй. “Програмчлалын арга зүй” хичээлээр өмнө нь кодны үлгэр загварын талаар үзсэн. Иймэрхүү олон янзын үлгэр загвар байдаг бөгөөд тэдгээрийн гол санааг тогтоох авах нь чухал байдаг. Button Demo-ийг уншсан хэсэгт санал санал болгосны дагуу импортлож, орчуулаад, ажиллуулна. Мөн дээрхтэй адил ерөнхий үлгэр болох загварыг тогтоохоос гадна, хэрхэн ажиллаж буйг сайн анхаарна уу. Шинээр нэг товчлуур нэмж ойлгосон эсэхээ шалгана уу. Ялангуяа товчлуурын нэг объекыг шинээр үүсгэх, түүний гарааны төлөв байдал болон харагдацыг өөрчлөх, сонсуур (listener) болгож бүртгэх, мөн түүний GUI-ийн чингэлэг рүү товчлуурыг нэмэх зэрэг үлгэрийг олж илрүүлэн тогтоогоорой. Товчлуурын actionPerformed() гэх үзэгдэл боловсруулагч (event handlers)-ийг хэрхэн кодлосныг бас сайн хараарай. Гүйцэтгэх ажил: Шинэ төсөл (project) үүсгэж, hbbv2.x –ийн эх кодыг импорт хийгээд, түүнийг гурван товчлуур харуулдаг болгож өөрчилнө. Эдгээр нь тус бүр “Pause”, “Resume”, “Single Step” гэсэн нэртэй байна. Гурван товчлуурын үйлдэл зохицуулагч (action handlers) нь тоглоомыг түр зогсоох, үргэлжлүүлэх эсвэл нэг-алхам зэргийг одоогоор хийхгүй ч, дараахыг хийдэг байна. Үүнд: • Товчлуурыг JPanel –ийн толгой хэсэгт цувруулан байрлуулна. • Үзэгдэлд тохирсон шошго наана (ж.нь түр зогсоох товчлуурын хувьд “Pause”). • Гарын тохирсон товчилборыг олгоно (ж.нь түр зогсоох товчлуурын хувьд “P”). • Эхэндээ Resume товчлуур идэвхгүй, Pause дарагдсаны дараа идэвхжинэ. • Эхэндээ Pause товчлуур идэвхжинэ, Pause дарагдсаны дараа идэвхгүй болно. 6. (6 оноо) Java Tutorials –ийн хичээл How to Use Swing Timers болон javax.swing.Timer гэсэн хуудсыг уншина. Програм хангамжийн хөгжүүлэлт ICSI402 Гүйцэтгэх ажил: Huckle Buckle Beanstalk тоглоомын түр зогсоох, үргэлжлүүлэх эсвэл нэг-алхам зэрэг үйлдлийг зохистойгоор гүйцэтгэхийн тулд өөрийн гурван товчлуурын actionPerformed() боловсруулагч дотор тоглоомын цаг (game timer) –ийн stop(), start() болон setRepeats(false) гэсэн аргыг дуудаж өгнө. Оноог бүрэн авахын тулд програм нь дарагдах шаардлаггүй товчлуурыг идэвхгүй болгосон байна. Ж.нь ажиллаж байх үед, “Resume” болон “Single-Step” идэвхгүй байна. Дээр тодорхойлсон шаардлагыг хэрэгжүүлэх үед өөрчилсөн эсвэл нэмж бичсэн аргын код, түр зогсоох анх удаа дарагдахаас өмнөх болон дараах байдлыг харуулсан цонхны хоёр зураг зэрэг зүйлсийг илгээнэ үү.

private static void createAndShowGUI() {

//Create and set up the window.

JFrame frame = new JFrame("HuckleBuckle viewer");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setPreferredSize(new Dimension(500, 500));

//Create a menu bar with three buttons. Make it have a green background.

JMenuBar greenMenuBar = new JMenuBar();

greenMenuBar.setOpaque(true);

greenMenuBar.setBackground(new Color(154, 165, 127));

greenMenuBar.setPreferredSize(new Dimension(120, 40));

greenMenuBar.add(new HbbButtons(greenMenuBar.getBackground()));

frame.setJMenuBar(greenMenuBar);

frame.getContentPane().add(new HuckleBuckle(), BorderLayout.CENTER);

//Display the window.

frame.pack(); // it'll be just big enough for the preferred sizes of its components

frame.setVisible(true);

}

/\*

\* This class is derived from ButtonDemo.java in the Java Tutorials

\*

\* I have overridden paintComponent(), so that these buttons will honour the

\* background colour of the JMenuBar in which they are embedded. Without

\* the override, the background colour may be defined by the look-and-feel, see

\* https://docs.oracle.com/javase/8/docs/api/javax/swing/JComponent.html#setBackgroundjava.awt.Color-

\*

\*/

@SuppressWarnings("serial")

public class HbbButtons extends JPanel

implements ActionListener {

protected JButton bPause, bResume, bSingleStep;

private Color backgroundColour;

/\*\*

\* Creates three buttons, and adds them to this JPanel

\*

\* @param bg

\* background colour

12

\*/

public HbbButtons(Color bg) {

backgroundColour = bg;

bPause = new JButton("Pause", null);

bPause.setVerticalTextPosition(AbstractButton.CENTER);

bPause.setHorizontalTextPosition(AbstractButton.LEADING); //aka LEFT, for left-to-right

locales

bPause.setMnemonic(KeyEvent.VK\_P);

bPause.setActionCommand("pause");

bPause.setOpaque(true);

bResume = new JButton("Resume", null);

bResume.setVerticalTextPosition(AbstractButton.BOTTOM);

bResume.setHorizontalTextPosition(AbstractButton.CENTER);

bResume.setMnemonic(KeyEvent.VK\_R);

bResume.setActionCommand("resume");

bResume.setEnabled(false);

bResume.setOpaque(true);

bSingleStep = new JButton("Single Step", null);

//Use the default text position of CENTER, TRAILING (RIGHT).

bSingleStep.setMnemonic(KeyEvent.VK\_S);

bSingleStep.setActionCommand("step");

bSingleStep.setEnabled(false);

bSingleStep.setOpaque(true);

//Listen for actions

bPause.addActionListener(this);

bResume.addActionListener(this);

bSingleStep.addActionListener(this);

bPause.setToolTipText("Click this button to pause the simulation.");

bResume.setToolTipText("Click this button to resume the simulation.");

bSingleStep.setToolTipText("Click this button to allow a single move.");

//Add Components to this container, using the default FlowLayout.

add(bPause);

add(bResume);

add(bSingleStep);

}

/\*

\* The game's button events are handled here

\*/

public void actionPerformed(ActionEvent e) {

switch (e.getActionCommand()) {

case "pause":

bPause.setEnabled(false);

bResume.setEnabled(true);

bSingleStep.setEnabled(true);

break;

case "resume":

bPause.setEnabled(true);

bResume.setEnabled(false);

bSingleStep.setEnabled(false);

break;

case "step":

}

}

@Override

13

protected void paintComponent(Graphics g)

{

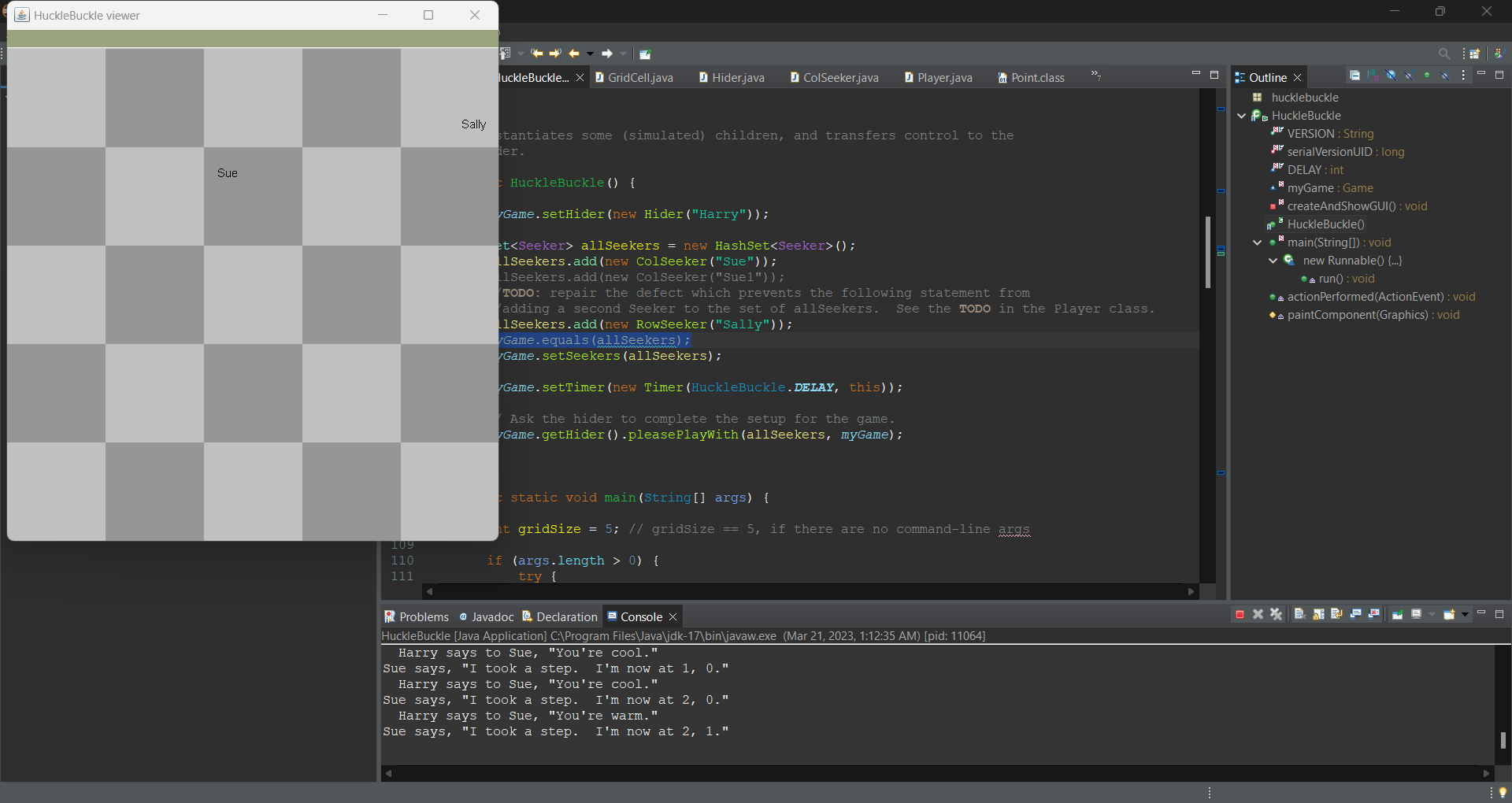
super.paintComponent(g);

g.setColor(backgroundColour);

g.fillRect(0, 0, getWidth() - 1, getHeight() - 1);

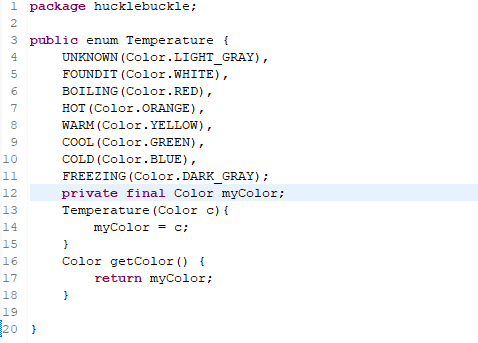
}

}

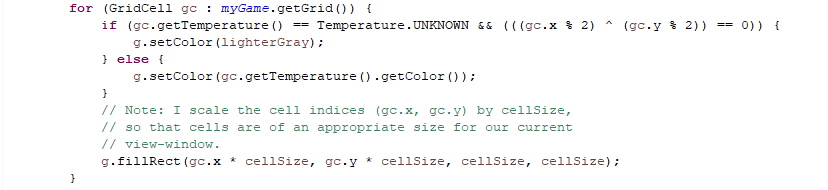


**Бүлэг даалгавар 5 : enum-д инстанцийн хувьсагч, арга нэмэх мөн зурах**

7. Өмнөх хувилбарын Temperature enum–ийг түүний инстанц болгонд өнгө (*Color*) холбож өгөн өөрчилж мөн температурыг хайгчид хэлж өгсөний дараа *HuckleBuckle* дах хэрэглэгчийн бичсэн зурах кодыг дөрвөлжин бүрт температурт тохирсон өнгөөр буддаг болгож өөрчилж, *HuckleBuckle* –ийн шинэ хувилбарыг хөгжүүлнэ үү. Өөрчлөлт хийсэн болон нэмсэн бүх код, Temperature enum, температурын тайлан мэдээний өөрчлөлт.



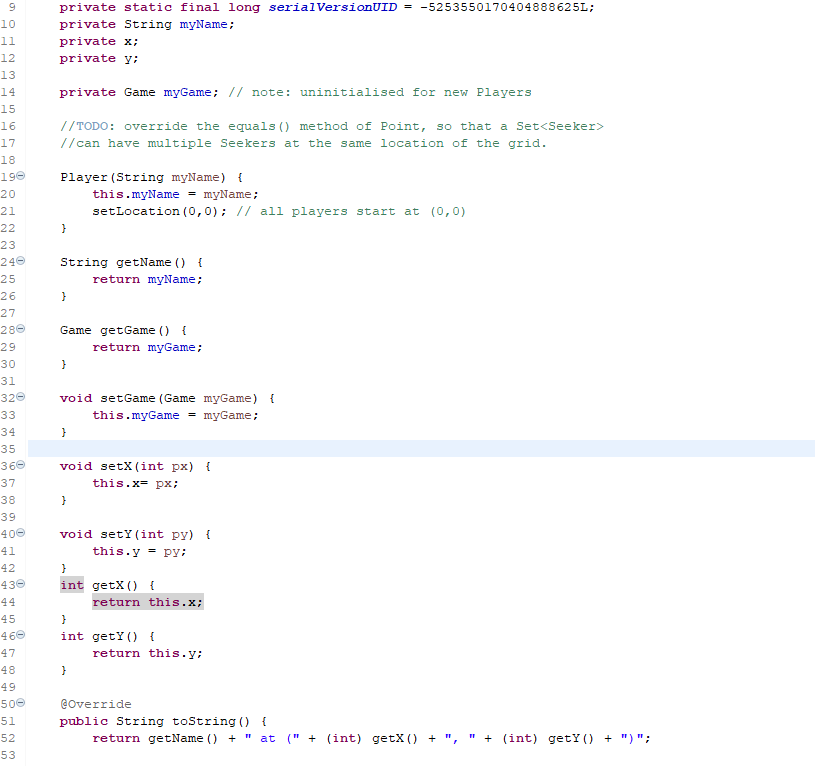
Тухайн хэсгийн Temprature-т тохирсон өнгөөр будах.



**Бүлэг даалгавар 6 : Тоглогч цэгтэй байх уу эсвэл цэг үү?**

8. **(4 оноо)** *Гүйцэтгэх ажил: HuckleBuckle* –ийн суурь кодонд чухал өөрчлөлт хийх зохистой байдлын талаар авч үзье. Тоглогч бол цэг гэхээсээ цэгтэй байх гэдэг нь илүү тохиромжтой байх. Хэрэв нийлмэл боломжтой бол удамшилыг ашиглах нь ерөнхийдээ муу санаа гэж хичээлээр үзсэн. Temperature enum, температурын тайлан мэдээний өөрчлөлт, торон нүдний ялгаатай өнгийг харуулсан дэлгэцийн зураг зэргийг илгээнэ. Тоглогч нь цэгтэй байх гэсэн *HuckleBuckle* –ийн шинэ хувилбарыг үүсгэн, зохиомжийн энэ шийдлийн талаар туршилт хийнэ үү. Өөрчилсөн кодын мөр бүрийг бичиж тэмдэглэнэ үү. Энэ зохиомжийн өөрчлөлтийг хийх үед тулгарсан асуудлын талаар үнэлсэн, давуу сул талыг харуулсан тайлбар илгээнэ үү. Нийлмэл нь удамшилаас ямар чухал давуу, сул талтай, өөрчлөлт хийхэд гарсан хүндрэл (ж.нь их хэмжээний код өөрчлөх)

Player class-н Points-оос удамшиж буй удамшлыг байхгүй болгосон ба Player class-д өөрийг нь хаана байгааг илэрхийлдэг x, y хувьсагч нэмж өгч цэг болон тоглогчийн харьцааг нийлмэл буюу Тоглогч байхгүй бол түүнийг илэрхийлсэн цэг байхгүй болгосон. Мөн тоглогчийн координат буюу x, y-т хандах зорилгоор getter setter функцууд бичиж өгсөн. Нэгэнт хэрэгжүүлсэн хэрэгжүүлэлтэнд player-н координаттай харьцах бүх хэсэгт өөрчилсөн өөрчлөлтөөрөө хэрэгжүүлэх нь бага зэрэг төвөгтэй байдал үүсгэсэн. Нийлмэл харьцаа нь илүү хүчтэй холбоос үүсгэдэг ба player байхгүй бол цэг байхгүй гэх утгыг илэрхийлэх бол удамшил хийсэн тохиолдолд тоглогч байхгүй ч цэгийг үүсгэх боломжтой гэдгээрээ давуу талтай. Player-ээс гадна цэгийг давхар ашигладаг тохиолдолд удамшил ашиглах нь хэрэгжүүлэлтийн хувьд хялбар. Харин цэгийг Player-ээс гадна ашиглахгүй байгаа тохиолдолд нийлмэл холбоос нь хялбар бөгөөд оновчтой хэрэгжүүлэлт юм.



**(6 оноо)** *Гүйцэтгэх ажил:* Нэгэн хайгч өөр нэгэн хайгчтай адилхан нүд рүү нүүх бол (эхлэх нүд *cell (0,0)*-ийг оролцуулахгүйгээр) түүний translate() гэсэн аргыг гүйцэтгэх болгожгүй болгож тоглоомын дүрмийг тохируулна уу. Хайгч нь өөр нэг хайгч нүдийг чөлөөлөх хүртэл MOVING төлөвт урагшлахгүй байж байна. Тоглоомд зөвхөн нэг л ялагч байна (объектыг эхлэх нүдэнд нуугаагүй л бол).

* *MOVING* төлөв рүү тоолуур нэмж өгөх хэрэгтэй, ингэснээр хайгч нүүж чадахгүй удвал *QUITTING* төлөв рүү шилжих болно (8 timer-events ч юм уу).
* Програмыг олон хайгчтай тестлэх хэрэгтэй. Тиймд хоёр дах комманд-мөрийн аргумент шаардлагатай (гарааны утга нь 2, ихдээ 10, багадаа 1).
* Дараагийн очих газрыг олохдоо урьдчилан тооцоолдог, шинэ төрлийн хайгчийг нэмж өгөх хэрэгтэй. *ASKING* төлөвт температур нь хэдийн мэдэгдсэн ямар нэгэн нүд рүү орох хэрэггүй.
* Олон хайгчтай тоглоомд хайгч нь янз бүр байх хэрэгтэй. Иймд коммандын мөрд өгсөн хэрэглэгчийн тоогоор эхлээд ухаалаг, дараа нь мөрөөр хайгч, дараа нь баганаар гэсэн дарааллын дагуу дахин давтаж хайгчийн объектыг үүсгэнэ.
* Хайгч нь түүний одоогийн байрлал, төлвийг оношлож болохуйц хэрэгтэй мэдээллийг консолын гаралт руу өгөх хэрэгтэй.
* Найм дугаар даалгавар гүйцэтгэхээс өмнө хөгжүүлсэн код дээр хөгжүүлж болно.
* Өөрчилсөн хайгчийн аргын код, “ухаалаг” хайгчийн аргын код, кодны эцсийн хувилбарын консол болон *GUI* –ийн гаралтын жишээ, програм шалгасан байдлыг харуулсан, зэргийг илгээнэ үү.

