JAVA25, JAVAD25, OOPJ
Inlämningsuppgift 3.

Ett 15-spel är ett spel med 15 brickor fördelade på 4 rader och 4 kolumner. En plats är alltid ledig. Spelet går ut på att brickorna blandas, och sedan sorteras i rätt nummerordning genom att de dras, en i taget, in till den tomma platsen (som då kommer att hamna på ett nytt ställe) och på så sätt flyttas runt.

Er uppgift är att bygga ett digitalt 15-spel i Java.

Betygskriterier:

För att bli **Godkänd (G)** krävs följande:

- Ett grafiskt gränssnitt som visar de numrerade brickorna i spelet.
- När man klickar på en bricka bredvid den tomma platsen ska den klickade brickan byta plats med ”tomma platsen”.
- Om en klickad bricka **inte** ligger bredvid tomma platsen ska de inte byta plats.
- Det ska finnas en knapp, ”Nytt spel”. När man klickar på denna ska alla brickorna blandas i slumpmässig ordning.
- När spelet är löst (alla brickor ligger i nummerordning) ska ett meddelande ”Grattis, du vann!” visas på skärmen.
- Programmet ska funka!
- Koden ska vara enkelt läsbar och prydligt skriven.
- Bägge personer ska vara med och skriva kod (förutsatt att ni gör uppgiften parvis).
- Koden ska laddas upp på StudentPortalen (det räcker med att 1 pers/grupp laddar upp)
- En länk till koden, på GitHub, ska finnas inlagd på Portalen.

För att bli **Väl Godkänd (VG)** måste lösningen uppfylla följande:

- Alla krav för att få G ska vara uppfyllda.
- Ni ska ha jobbat i feature-brancher i git, som successivt mergats in i master-branchen, när ni skrev er kod.
- Vid redovisningen ska branchning och mergning framgå från historiken i git eller GitHub.
- Ni får bonuspoäng om ni kan berätta om vilka horribla mergekonflikter ni har haft - och löst

När brickorna blandas slumpvis kan de hamna i en ordning som gör att spelet inte går att lösa. **Ni behöver inte ta höjd för att fixa eller detektera detta** (det finns algoritmer på nätet som detekterar problemet, ni kan klippa in en sådan om ni vill, lägg i så fall in en länk som kommentar till vart ni hittade den).

Denna uppgift görs gärna parvis – detta för att ni ska få erfarenhet av hur det är att vara flera som mergar och branchar i git innan ni börjar med grupparbetet. Ni ansvarar själva för att bilda par. För dem som absolut inte vill jobba i par går det bra att jobba enskilt, men jobba ändå med branchning och mergning. Se till att era mergningar och branchningar syns i gitHub, man kan göra detta på olika sätt och på vissa sätt kommer det inte att synas, andra inte. Tänk på detta i tid!!!

Varje redovisning får 10 minuter, så gå noggrant igenom eran redovisning innan och se till att ni kan klocka av den på utsatt tid. Max 5 redovisningar per tids-slot. 1 redovisning/par.

När ni redovisar, visa upp:

- Att spelet startar och fungerar på specificerat sätt (att man kan flytta brickor etc.)
- Vad som händer när man vinner (ett tips är att bygga en specialinställning där alla brickorna läggs ut sorterade. Sen kan man flytta en bricka och sen flytta tillbaka den, så har man ”vunnit”)
- Visa koden snabbt
- Visa hur ni jobbat med mergning och branchning genom att visa upp din mergenings- och branchnings-historik i GitHub (i ditt GitHub-repo, välj Insights -> Network)
- Bägge personer i ett par måste prata/demonstera

För er som tycker att uppgiften är lätt finns här några förslag på utökning (ingen av dessa är nödvändiga för att få G eller VG, ni gör dessa bara om ni vill)

- Lägg till funktionalitet så användaren kan välja hur många rader och kolumner spelet ska ha.
- Lägg till funktionalitet så att användaren kan byta färger på spel och brickor.
- Lägg till funktionalitet så att du kan flytta flera brickor i taget. T.ex: om användaren klickar på en bricka som ligger två brickor till vänster om ”tomma platsen” flyttas bägge dessa brickor. Bara brickor som ligger direkt i linje med ”tomma platsen” (och inte diagonalt) ska kunna flyttas på detta sätt.
- Lägg till funktionalitet så att ni räknar hur manga drag en användare tar på sig för att lösa spelet.
- Lägg till funktionalitet för att mäta hur lång tide en användare tar på sig att lösa spelet.
- Gör en highscore-lista över vilka användare som har löst spelet snabbast eller med minst antal drag. För att detta ska vara möjligt måste ni även hålla reda på olika användare, så funktionalitet för detta måste då läggas till.

Om den kod ni visar upp har några som helst likheter med de lösningar som flyter runt på Internet blir ni omedelbart underkända och får göra en annan, mycket svårare, garanterat supertråkig uppgift. Detta gäller i synnerhet om er lösning innehåller klassen Graphics2D.

Som vanligt är det förbjudet att använda ai-genererad kod.

Lycka till!