Jag valde att använda en abstrakt ship klass, som tillåt mig att definiera attributen med constraints och metoder som skulle ärvas av varje efterföljande sub klass. Eftersom det inte fanns förväntningar på en typ av ship klass som inte delar dessa gemensamma attribut, kände jag att det var onödigt att skapa ett interface. Om en ny typ av ship skulle introduceras (till exempel en ship som hade kapacitet att vara på diagonalen på spelplan), då skulle jag tillägga ett interface Iship, implementerat av olika abstrakt ship klasser.

För att kunna ha två spelare, två spelplaner skulle behöva skapas, och extra boundary klasser som ger spelarna möjlighet att placera sina ships själv i början av spelet. Sedan en extra panel för att hålla den array av knappar som representerar den andra spelplan. Huvudvyn kan helt enkelt visa de två spelplan sida vid sida, och inaktivera / aktivera varandra efter varje spelares gissning. Jag skulle föreslå att model klassen skapar två olika spelplaner, som innehåller egna rutor. Model klassen kunde då ha modifierat metoder, som pekar till den ena eller andra spelplan beroende på vems tur det är.

Vad gjorde ni för att programmet ej skulle hänga, om det inte kunde lista ut hur man passar alla ships på spelplan?

Hur skapar programmet en ny storlek av spelplan och nya highscores, utan att förlora de gamla highscores från de andra storlekarna?

För att en spelare kan spela mot en AI-spelare, skulle programmet kunna omstruktureras enligt ovan för två spelare, men spelar tvås tur utförs av en AI istället. Den kan utlösas av ett enkelt boolean attribut som anger om programmet ska vänta på mänsklig input eller få AI-klassen att skapa den nödvändiga indata.