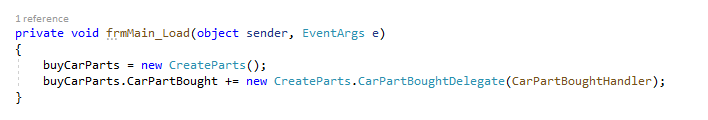
1. Uzraksti **piecas** .NET ietvara bāžu klašu bibliotēkas sniegtās priekšrocības?
   * 1. Bāzēts uz objektorientētas programmēšanas, kas ļauj software sadala vairākās mazākās daļās.
     2. Vizuāli ērti un viegli rediģēt izskatu, vienlaikus izmainot arī kodu.
     3. Mazāk problēmas, palaižot .NET aplikāciju citās platformās un operētājsistēmās (C#).
     4. Viegli uzturams kods, sistēma, aplikācija.
     5. Iebūvētas debugging iespējas.
2. Uzraksti **piecas**.NET ietvara (*parastās*) klases priekšrocības pār struktūras klasi?
   * + 1. Atbalsta mantošanu (inheritance);
       2. Ir pointera (references) tips;
       3. Reference var būt null;
       4. Atmiņas rezervēšanas priekšrocības;
       5. Netiek padotas kā vērtība, atšķirībā no struktūras klases
3. Nosauc kādus **piecus** parametrus ietver .NET Windows formu komponente ‘*TextBox*’?
   * 1. TextAlign
     2. UseWaitCursor
     3. ScrollBars
     4. BorderStyle
     5. Font
4. Kas .NET ietvarā ir delegāts, paskaidro tā pielietošanas veidu (izejot no **viena** piemēra)?

Delegāts norāda uz funkciju un spēj padot funkciju kā parametru. Var arī teikt, ka delegāts ir .NET objekts, kas norāda uz metodi, kas sakrīt ar konkrētu atgrieztās vērtības tipu.

Piemērs no 3. mājas darba:



1. Nosauc **piecas** raksturīgākās kļūdas, ko pieļauj izstrādātāji veidojot .NET ietvarā Windows formu lietotnes interfeisu jeb saskarni?
   * + 1. Sarežģīts izkārtojums, tādejādi apgrūtinot lietotāja navigāciju.
       2. Slikta lietotāja asistence, kad tiek pieļautas kļūdas (Nepietiekams vai nesaprotams kļūdas paziņojums)
       3. Klases vai komponentes, kas nav pareizi un adekvāti nosauktas (Piemēram *txtLabel* vietā ir *TxTLABEL*).
       4. Bezjēdzīga informācija, kas ir bez konteksta un piesārņo kopējo skatu.
       5. Slikta un nepaskaidrota formas vai kādu citu input lauku validācija.
2. Apraksti, kāda .NET ietvarā ir mainīgā redzamie apgabali (*piecus*)?
   1. Public
   2. Protected
   3. Internal
   4. Protected internal
   5. Private
3. Kas .NET ietvarā Windows formu lietotnes izstrādnē ir funkcionalitāte, ko sauc par ‘*Option Strict*’, ko tā dara un kādai .NET ietvara programmēšanas valodai tā pieder?

Funkcionalitāte “Option Strict” neļauj nejauši konvertēt vienu datu tipu uz otru, kas ir mazāk precīzs (piemēram no integer uz byte). Šo funkcionalitāte ir iespējams arī izslēgt, taču noklusējuma uzstādījumos šī iespēja ir ieslēgta. Šī funkcionalitāte pieder pie Visual Basic programmēšanas valodas.

1. Apraksti, kā .NET ietvarā notiek programmētāja uzrakstītās sintakses kompilēšana (ņemot vērā, ka nav sintakses kļūdas)?

.NET ietvarā par programmētāja uzrakstītās sintakses kompilēšanu atbild JIT compiler, kas ir daļa no CLR (Common Language Runtime). CLR nodrošina visu, kas saistīts ar .NET aplikācīju palaišanu. Kompilācija notiek tikai 1 reizi un pēc palaišanas, saglabājas ietvarā kā binārais kods.

1. Kas .NET ietvarā Windows formu lietotnē ir skaidrā datu konvertācija, uzraksti **piecus**dažādus skaidrās konvertācijas piemērus?

**1.** Long longLong = 888;

Int intInteger;

intInteger = (int)longLong;

**2.** Long longLong = 888;

Short shortShort;

shortShort = (short)longLong;

**3.** Single singleSingle = 888;

Char charChar;

charChar = (char)singleSingle;

**4.** Double doubleDouble = 888;

Float floatFloat;

floatFloat = (float)doubleDouble;

**5.** Int intInteger = 888;

short shortShort;

shortShort = (short)shortShort;

1. Paskaidro, ko nozīmē .NET ietvara Windows formu lietotnes izstrādnē jēdziens ‘*Boxing*’?

Jēdziens “Boxing” apraksta netiešo konvertāciju no vērtības tipa uz references tipu.

Piemērs: Int number = 8008135; Object objekts; objekts = number;

Tāpat kā notiek “Boxing”, tāpat arī “Unboxing”:

Piemērs: objekts = 8008135; number = (int)objekts;