Kassasystemet

Kassasystem – Objektorienterad kod

Allmänt

Programmera ett **kassasystem** – (som man har i kassan i en matbutik)

Fusk

Uppgiften ska lösas individuellt. Vid redovisningen får du frågor och måste beskriva hur ditt program fungerar så du måste kunna förklara hur all din egen kod fungerar.

Det är inte tillåtet att plagiera (skriva av) någon annans kod. Även om du byter namn på klasser, metoder och variabler är det enkelt att se om två olika applikationer har samma struktur.

Betyg

Det går att få G, VG eller IG på uppgiften. Den är obligatorisk tillsammans med andra uppgifter, examinerande laborationer och tentamen för att få G på hela kursen. För att få G på inlämningsuppgiften ska alla krav märkta med G vara uppfyllda. För att få VG på inlämningsuppgiften ska alla krav märkta med G och VG vara uppfyllda.

Redovisning

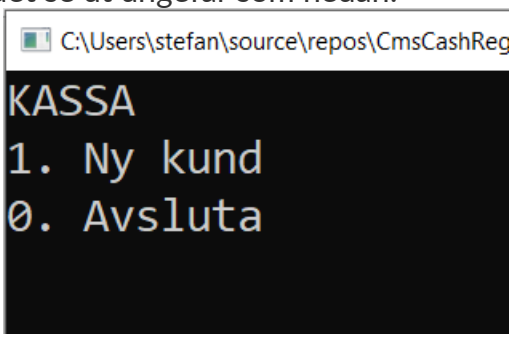
Skapa en privat github och bjud in mig som admin/collaborator (RichardChalk) så jag kan se din kod i repo(s). Det är där jag kopplar repo till person och betygsätter. Skriv in länk till ert repo i inlämningsuppgift på skolans portal - det är där jag kopplat repo till person och betygsätter. Jag ska kunna ladda ner er kod och trycka F5 så ska allt funka!

Exempel

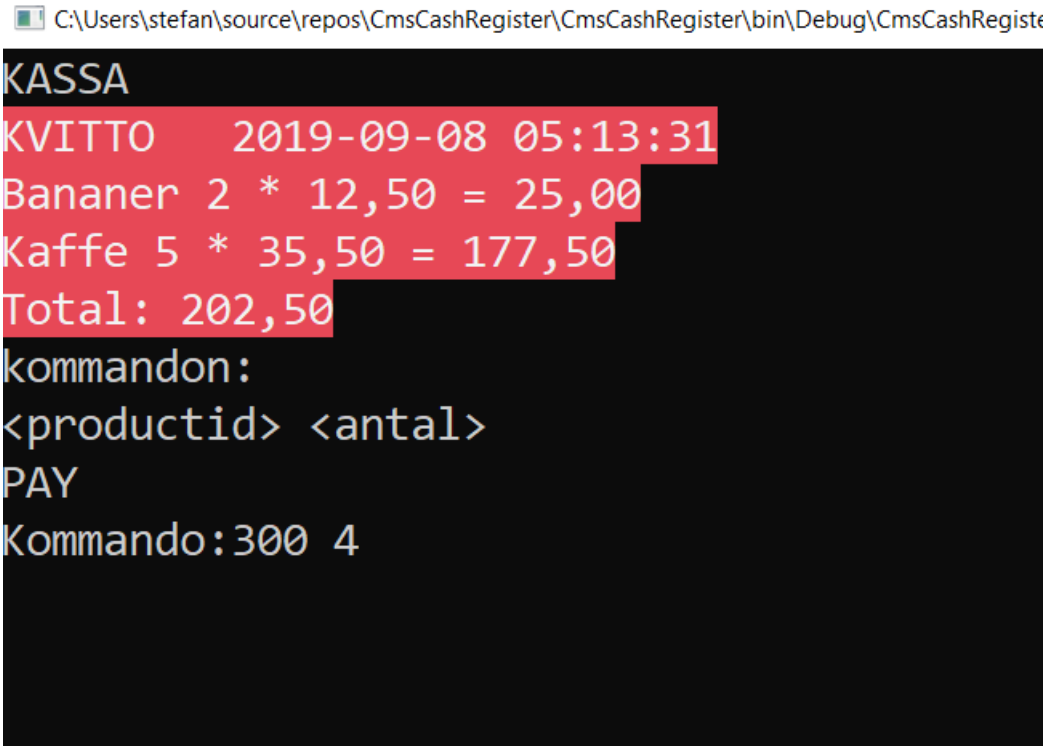
Produkter i kassasystemet lagras i en fil. Följande data ska lagras på Produkt:

- produktid (snabbkommando i kassan, ex ”300” för bananer nedan)
- produktnamn
- pris
- pris typ – är det per kilo eller per styck

När man kör kassan ska det se ut ungefär som nedan:



Vid val av **1. Ny kund** så startas kassan med en ny försäljning



Här finns två kommandon:

<produktid> <antal> ex 300 1, betyder lägg till en av produktid

PAY = vi ”fejkar” att det betalas och kvittot sparas ned (se nedan) och vi kommer tillbaka till menyn

Krav för godkänt

- Funktionalitet enligt ovan (OBS: Du **MÅSTE** följa specifikationen – tex inmatningen av produkt och antal **MÅSTE** ske enligt ”300 2” på en rad med mellanslag emellan)

- Funktionalitet enligt ovan (OBS: Du **MÅSTE** använda dig av Objekt Orienterad Paradigm – dvs. Klasser, Interface & objekt)

- Kvitton sparas ned vid **PAY** till en fil **RECEIPT_yyyyMMdd.txt** (dagens datum). OBS! Det blir alltså FLERA kvitton i samma fil. Fundera ut och implementera någon slags särskiljare så man kan skilja olika kvitton åt

Krav för VG

- Adminverktyg där man ska kunna ändra namn och pris för produkter
- Och dessutom lägga till nya produkter
- kvitton ska ju ha en ett LÖPNUMMER och detta måste plussas med ett hela tiden (även om ni stänger ner programmet ska sedan nästa kvitto få senaste kvittonumret + 1)
- Det ska finnas möjlighet att administrera KAMPANJPRISER. Det som menas är att tex från 2023-03-18 till 2023-03-19 kostar mjölken 10 kr
- Lägga till/ta bort kampanjer.
- OBS: En produkt kan ha MÅNGA kampanjer
- Naturligtvis ska detta pris gälla och visas på kvitton vid dessa tillfällen
- Ett UML diagram av din lösning