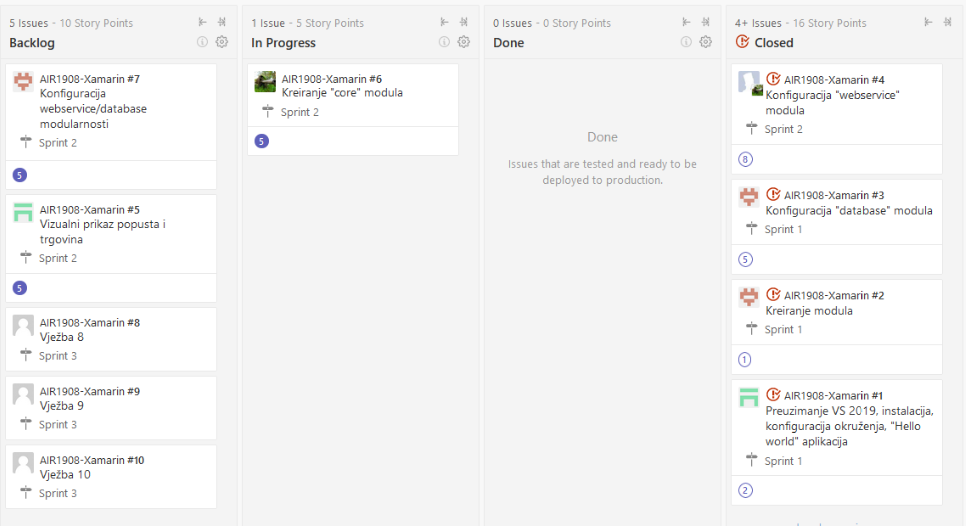
***Sažetak***

U ovoj vježbi konfigurirat ćemo core modul. U njega ćemo dodati nekoliko klasa koje će nam služiti za rad s podacima iz dva izvora: baza i web servis. S obzirom da želimo da naša aplikacija zna dohvatiti podatke iz bilo kojeg izvora, u ovoj ćemo vježbi napraviti sve što je potrebno kako bismo ostvarili takvu modularnost.

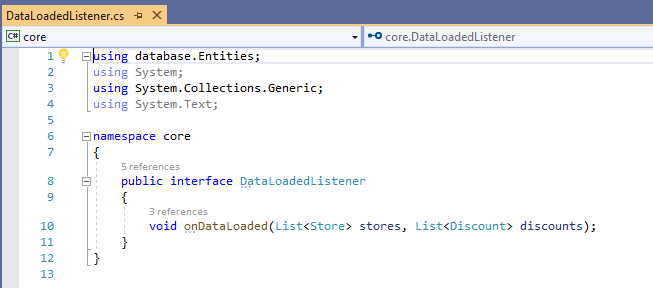
Zadnja verzija aplikacije: Commit b4ea28b [<https://github.com/AIR-FOI-HR/AIR1908-Xamarin>]

***Što radimo?***

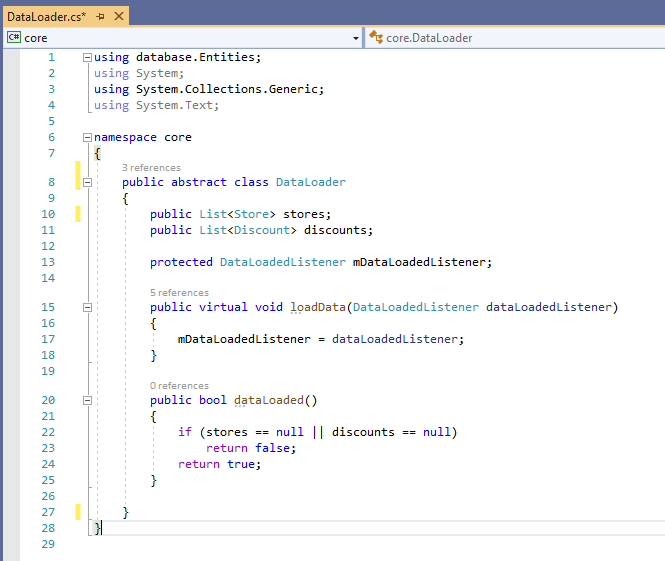


U prošloj smo vježbi dodali core modul, a u ovoj vježbi nadopunit ćemo taj modul potrebnim klasam i sučeljima (eng. *interface*) kako bismo ostvarili modularni dohvat podataka. Za početak, potrebno je u core modulu dodati referencu na database modul. Proširimo core modul, desni klik na Dependencies > Add Reference… Odaberemo database na prikazu modula.

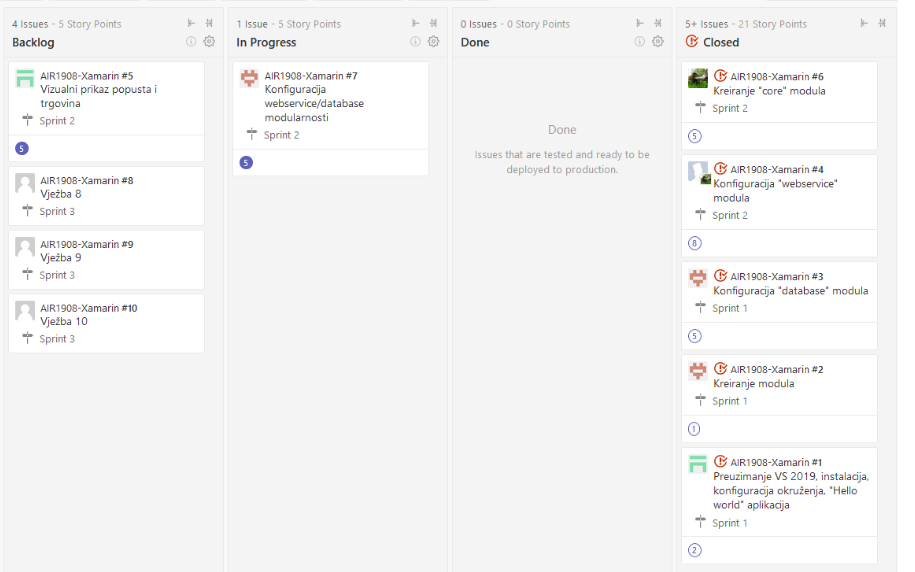
Prvo ćemo kreirati klasu DataLoadedListener koja će kreirati događaj (metodu) onDataLoaded kad podaci budu dostupni.



Sljedeća je na redu klasa DataLoader koja će biti zaslužna za dostavljanje podataka neovisno o izvoru. Bit će to apstraktna klasa, a s obzirom da imamo dva izvora podataka (web servis i baza), oba će naslijediti ovu klasu. U nju ćemo staviti konstruktor koji incijalizira osluškivač (eng. *listener*) i metodu dataLoaded koja provjerava jesu li podaci dostupni. Vraća *true* ako jesu, u suprotnom vraća *false*.

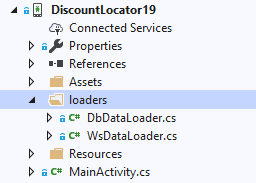


Sada smo pripremili core modul koji će nam poslužiti za dohvat podataka. Sljedeća je stvar omogućiti modularnost kod dohvaćanja podataka, tj. DataLoader mora nam dobaviti podatke neovisno o njihovom izvoru. Dakle, na redu je konfiguracija modularnosti dohvata podataka.

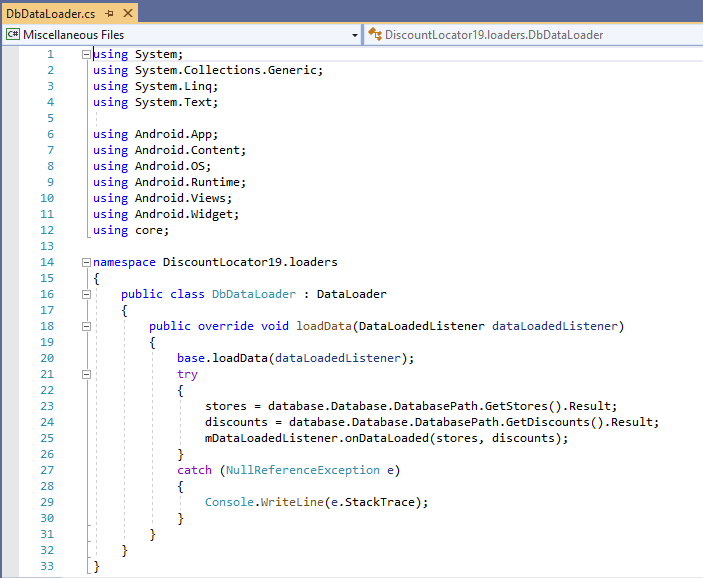


Do sada smo na dolazak podataka s web servisa samo implementirali ispis poruke u konzolu. S obzirom da je ova operacije asinkrona, podatke ćemo dobavljati preko osluškivača kojeg smo sada napravili (DataLoadedListener). Isto tako, podatke ćemo moći dobaviti iz baze pa ćemo i u tom slučaju koristiti DataLoadedListener. Prvi nam je korak kreirati dvije klase temeljene na DataLoader klasi koje će biti zaslužene za dohvat podataka. Jedna će se zvati DbDataLoader, a druga WsDataLoader. Po njihovim nazivima jasno je koja je klasa zadužena za koji izvor podataka.

Kreirat ćemo novu mapu loaders u DiscountLocator19 modulu. Desni klik na DiscountLocator19 modul > Add > New Folder > loaders. Desni klik na loaders mapu > Add > Class… > Name: DbDataLoader. Isto ponovimo i za WsDataLoader. Nakon toga, ovako bi nam trebao izgledati DiscountLocator19 modul.

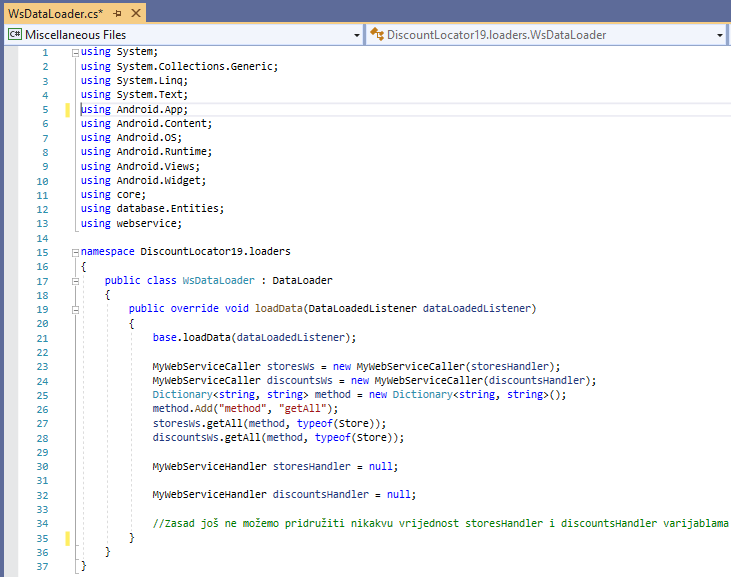


Sada ćemo implementirati DbDataLoader klasu. Nasljeđuje apstraktnu klasu DataLoader pa ćemo napraviti override metode loadData iz te klase. Ona prima DataLoadedListener tip podataka. Ključna riječ „base“ služi nam kako bismo pristupili članovima klase koju nasljeđujemo. U varijable *stores* i *discounts* spremamo liste dućana i popusta koje dohvaćamo iz baze ako ih ima. Zatim pozivamo onDataLoaded metodu iz DataLoadedListener klase i prosljeđujemo joj spomenute liste podataka.



Funkciju osluškivača omogućit ćemo tako da pozovemo njegovu metodu. Događaj dolaska podataka zapravo je poziv onDataLoaded metode iz klase koja će naslijediti DataLoadedListener interface. Kasnije ćemo vidjeti da će taj interface naslijediti klasa MainActivity jer ćemo podatke trebati ispisati na ekran.

Sada ćemo implementirati klasu WsDataLoader koja također nasljeđuje DataLoader klasu. I u ovom ćemo slučaju napraviti override metode loadData. U ovoj ćemo klasi ostvariti dva poziva web servisa: jedan za dohvat dućana, a drugi za dohvat popusta.



Kreirali smo dvije instance MyWebServiceCaller-a i svakom u konstruktor proslijedili pripadajući *event handler* (storesHandler i discountsHandler). Kreirali smo jedan rječnik (eng. *dictionary*) koji sprema tekstualne parove (ključ, vrijednost). Dodali smo jedan par: ('method', 'getAll') kojeg ćemo proslijediti metodi getAll. Na poziv ove metode dohvaćaju se podaci s web servisa, a kada su dohvaćeni poziva se metoda onDataArrived iz MyWebServiceHandler-a. Zasad ne možemo pridružiti nikakvu vrijednost varijablama storesHandler i discountsHandler jer nije moguće instancirati objekt klase MyWebServiceHandler. Ta nam je klasa *interface* i po definiciji ona se ne može instancirati. Zato ćemo napraviti novu klasu koja će nam služiti kao *factory[[1]](#footnote-1)* koji će razlikovati dućane od popusta i na temelju toga prosljeđivati podatke *handleru* za dućane ili *handleru* za popuste.

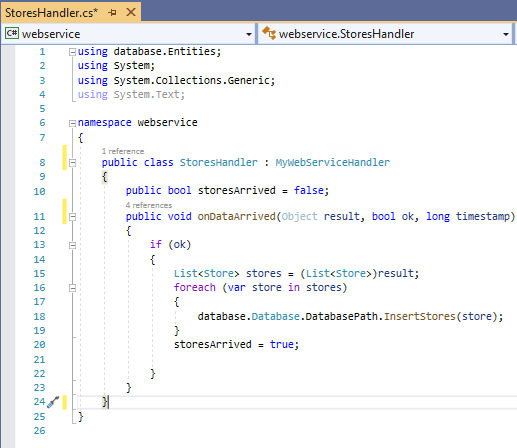


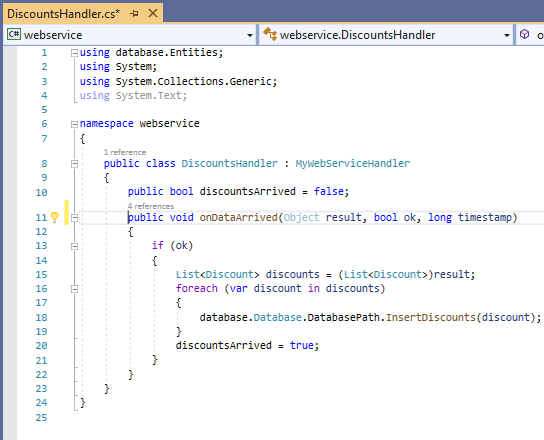
Proslijedili smo objekte dućana i popusta GetHandler metodi u našoj MyWebServiceHandlerFactory klasi. Kod pozivanja metode GetHandler proslijedili smo tip podataka Store i Discount. Sada ćemo kreirati *factory* klasu.



Metoda GetHandler prima generički tip podataka. To znači da smo tip podataka odredili kada smo instancirali objekt klase MyWebServiceHandlerFactory. Generički tip sadrži generički parametar T koji smo zamijenili s konkretnim tipom. Metoda provjerava koji smo tip podataka poslali i na temelju toga kreira novu instancu StoresHandler ili DiscountsHandler klase.

Sada ćemo kreirati StoresHandler i DiscountsHandler klase koje nasljeđuju MyWebServiceHandler *interface.*

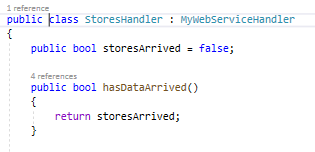


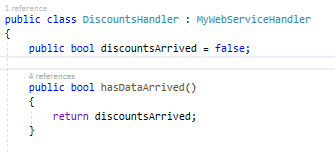


Možemo vidjeti kako obje klase nasljeđuju MyWebServiceHandler *interface* pa smo zato u obje napravili override metode onDataArrived. S obzirom da smo prilikom poziva web servisa pozvali metodu onDataArrived dva puta (jednom s listom dućana, a drugi put s listom popusta), upravljat ćemo s dva odgovora. Zasad ćemo dohvaćene liste dućana i popusta spremiti u bazu (iako to nije nužno).

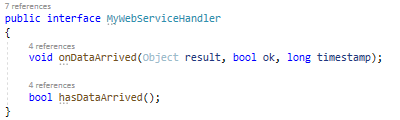
U ovim smo klasama kreirali i dvije varijable koje nam služe kao zastavice: storesArrived i discountsArrived. Ove su nam zastavice potrebne kako bismo znali u kojem su trenutku došli svi podaci s web servisa (i dućani i popusti). Možemo nastaviti tek kad su svi podaci došli (kad su obje zastavice postavljene na true). To ćemo iskoristiti u klasi WsDataLoader jer smo tamo i pozvali web servis.

Prvo moramo kreirati jednu metodu u klasama StoresHandler i DiscountsHandler koja će vraćati vrijednost zastavice.



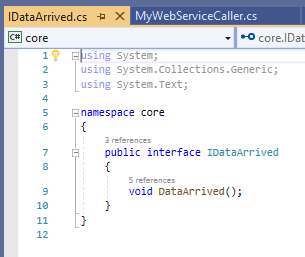


S obzirom da ove klase nasljeđuju MyWebServiceHandler *interface*, metodu hasDataArrived moramo dodati i u njega.

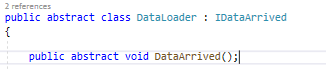


Ove ćemo metode pozivati u WsDataLoader klasi kako bismo povjerili stanje zastavica, tj. jesu li podaci s web servisa došli ili ne.

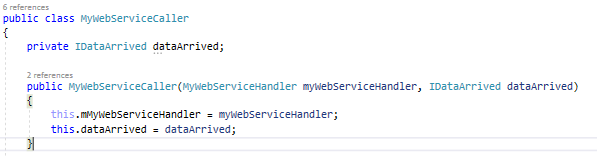
Prvo ćemo se vratiti u handleStores i handleDiscounts metode u MyWebServiceCaller klasu. Nakon što smo proslijedili deserijalizirane podatke onDataArrived metodi koja je postavila određenu zastavicu, ispitat ćemo vrijednost druge zastavice pozivanjem metode DataArrived. Ovu ćemo metodu kreirati u novom *interface­-u* IDataArrived u core modulu.



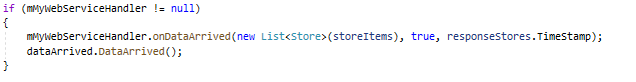
Ovaj *interface* nasljeđuje klasa DataLoader pa je potrebno dodati u nju istu metodu.

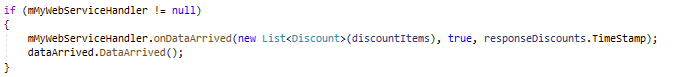


Sada ćemo promijeniti dodati varijablu dataArrived tipa IDataArrived u MyWebServiceCaller klasu i izmijeniti konstruktor te klase.

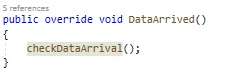


Tek sada možemo koristiti DataArrived metodu iz IDataArrived *interface-a*. To ćemo napraviti u metodama handleStores i handleDiscounts, nakon obrade podataka.

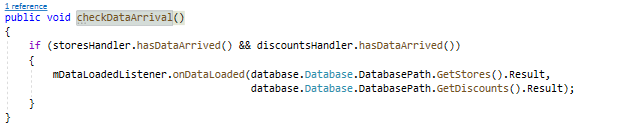




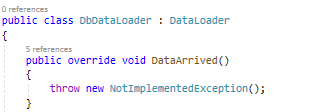
Ova metodu implementirat ćemo u WsDataLoader klasu kako bismo mogli nastaviti raditi s podacima s web servisa. Dakle, odemo u WsDataLoader klasu i dodamo DataArrived metodu.



Ona poziva funkciju checkDataArrival koja će provjeriti stanje zastavica. Ako su obje zastavice postavljene na *true*, osluškivač poziva onDataLoaded metodu kojo prosljeđuje dohvaćene podatke (trenutno se podaci dohvaćaju iz baze jer smo podatke s web servisa spremili u bazu).



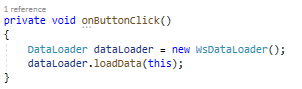
S obzirom da i DbDataLoader nasljeđuje klasu DataLoader u koju smo dodali metodu DataArrived iz IDataArrived *interface-a*, dopunit ćemo i tu klasu.



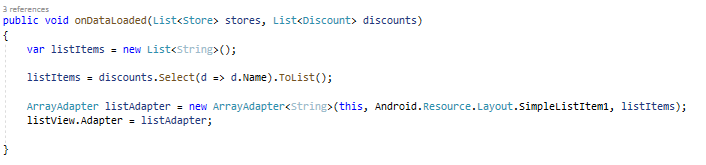
Sada ćemo se vratiti u MainActivity klasu. Ovdje ćemo omogućiti dohvat podataka. Dakle, ovdje nam treba osluškivač kojeg smo stvorili tako da će MainActivity klasa nasljeđivati klasu DataLoadedListener.



Dohvat podataka omogućit ćemo na klik gumba. U ovom smo slučaju pozvali WsDataLoader što znači da ćemo dohvatiti podatke s web servisa.

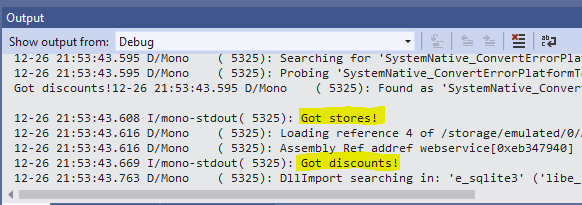


Kada podaci stignu, pozvat će se metoda onDataLoaded koja će nam dati liste dućana i popusta. Iz liste popusta uzet ćemo nazive i prikazati ih u listView elementu.

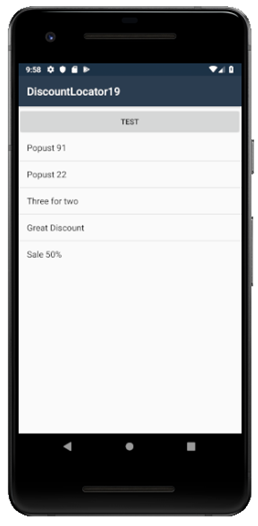


Sada je na redu demonstracija. Prije svega napravit ćemo *Rebuidl.* Onda ćemo pokrenuti emulator i vidjeti hoće li nam se na ekranu ispisati nazivi popusta.

U konzoli su nam se ispisale poruke o uspješnom dohvaćanju podataka s web servisa. To je znak da je naša aplikacija uspješno komunicira s web servisom.

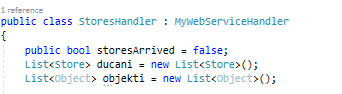


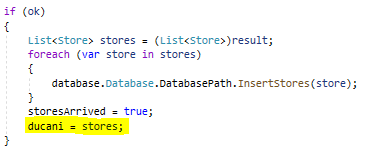
Emulator je na ekran ispisao sve nazive popusta. Vidimo da je ukupno 5 popusta, 2 iz baze i 3 s web servisa.



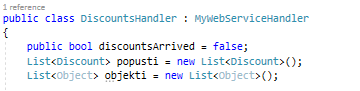
U klasama StiresHandler i DiscountsHandler podatke s web servisa spremili smo u bazu podataka. Zatim smo u WsDataLoader klasi u metodi checkDataArrival metodi onDataLoaded proslijedili sve što se nalazi u bazi. Međutim, mi želimo da se proslijede samo podaci s web servisa.

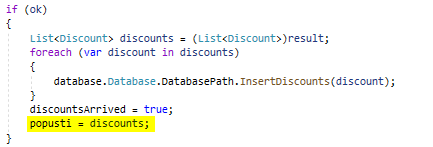
Zato ćemo prvo otići u StoresHandler i DiscountsHandler klase i dodati novu listu objekata u koju ćemo spremiti podatke dućane i popuste.



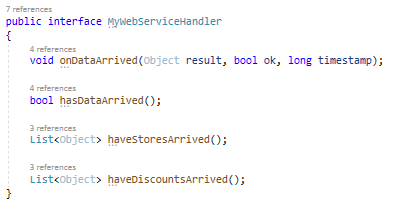


-----------------------------------------------------------------------------------------------------------------

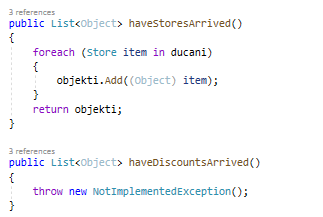




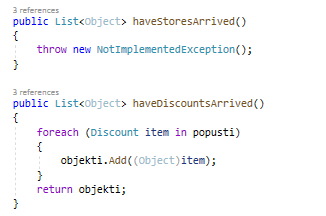
Zatim ćemo u ove klase dodati još jednu dvije metode koje će nam vraćati ove liste dućana i popusta onda kad ih budemo pozvali. Prvo ih moramo dodati u *interface* MyWebServiceHandler jer ga ove klase nasljeđuju.



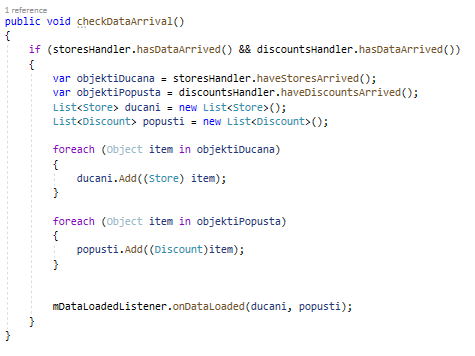
Sada te dvije metode možemo dodati u klase StoresHandler i DiscountsHandler. U StoresHandler klasi samo će nam haveStoresArrived vraćati podatke, a u DIscountsHandler klasi podatke će vraćati metoda haveDiscountsArrived.



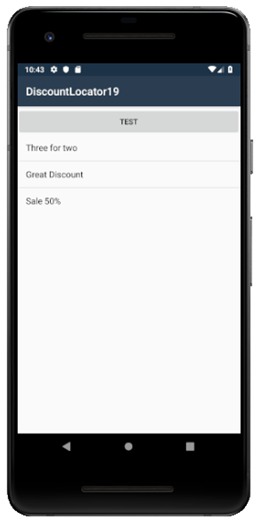
-----------------------------------------------------------------------------------------------------------------



Sada možemo izmijeniti checkDataArrival metodu u WsDataLoader klasi. Metodi onDataLoaded proslijedit ćemo samo one dućane i popuste koje smo dohvatili s web servisa. Dućane i popuste dohvatimo pozivajući metode haveStoresArrived i haveDiscountsArrived koje vraćaju listu objekata. Objekte ćemo samo promijeniti u željeni tip podataka.



Sad ćemo ponovno napraviti *Rebuild* našeg projekta i pokrenuti emulator.



Vidimo da nam aplikacija ispisuje samo popuste koje je dohvatila s web servisa. Ako želimo da nam ispiše samo popuste koji se nalaze u bazi, sve što trebamo napraviti je promijeniti jednu liniju koda u MainActivity klasi. Umjesto WsDataLoader, stavit ćemo DbDataLoader.





Na ovaj smo način ostvarili modularni dohvat podataka. U sljedećoj vježbi fokusirat ćemo se na korisničko sučelje. Napravit ćemo ljepši prikaz trgovina i njihovih popusta.

1. *Factory* uzorak dizajna - <https://www.tutorialspoint.com/design_pattern/factory_pattern.htm> [↑](#footnote-ref-1)