Upotreba dizajn šablona u Androidu – *Factory* i *Singleton*

Dizajn šabloni mogu biti implementirani na različitim platformama i u mnogim programskim jezicima. Šabloni omogućavaju opšta rešenja koja mogu biti ponovo iskorišćena u različite svrhe. Šabloni nisu skup pravila, već skup oprobanih rešenja problema. Pored upotrebe postojećih, možete i sami kreirati dizajn šablone.

Neke od prednosti upotrebe dizajn šablona:

- Uvodi se nivo apstrakcije koji olakšava modifikovanje koda koji se razvija. Postoje određeni šabloni koji su upravo dizajnirani za ovakve slučajeve.
- Mogu biti primenjeni na više nivoa, kako na opštu arhitekturu projekta, tako i na projektovanje prostih objekata.
- Doprinose manjoj količini komentara u kodu i propratne dokumentacije, jer i sami služe kao vid opisa. Ime klase ili interfejsa može poslužiti kao dovoljno objašnjenje svrhe unutar šablona.

Šabloni se mogu svrstati u tri kategorije:

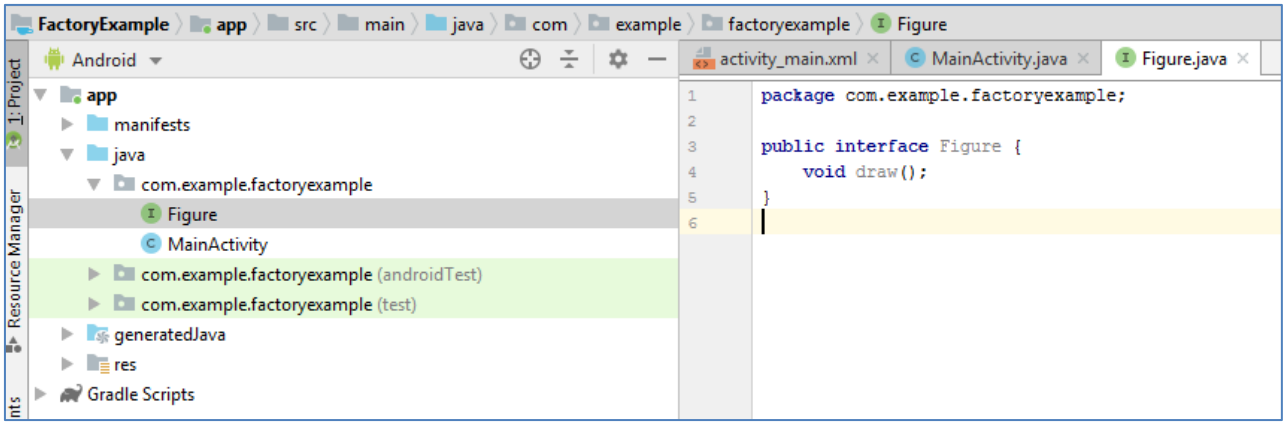
1. Šabloni za **kreiranje** objekata
2. Šabloni za **organizovanje** grupa objekata
3. Šabloni za **komunikaciju** između objekata

Factory šablon (engl. *Factory pattern*)

Jedan od najčešće korišćenih šablona u Java programskom jeziku. Spada u tip šablona za kreiranje objekata. Pruža jedan od najboljih načina za kreiranje objekata. U okviru ovog šablona, objekat se kreira ne otkrivajući logiku korisniku.

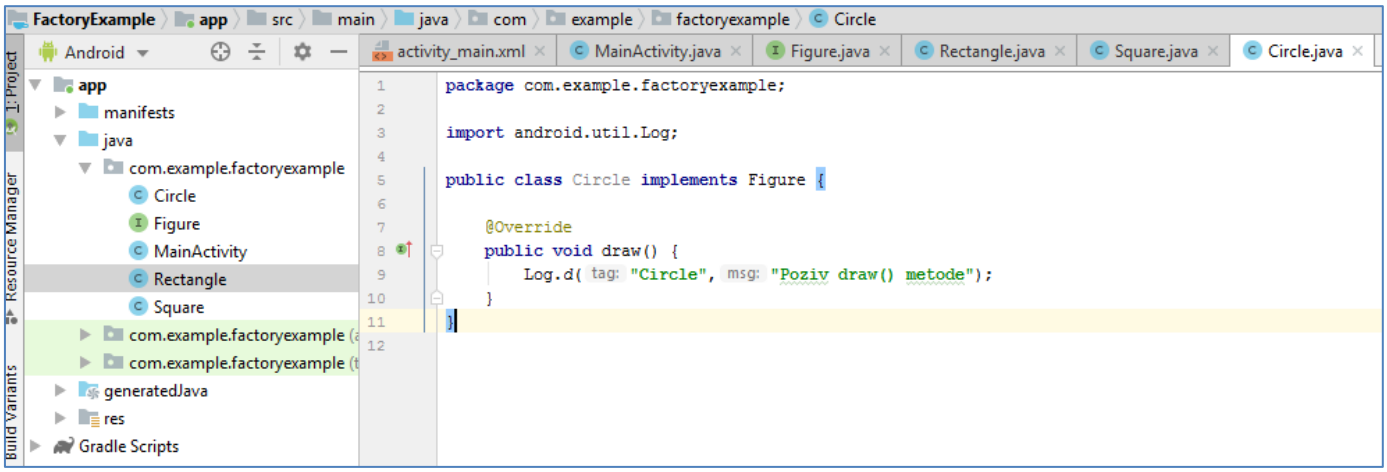
Primer

- Kreiramo nov Android projekat, FactoryExample.
- U okviru projekta, desnim klikom na MainActivity datoteku, biramo **New->Java class**. Za tip ove datoteke, izabrati opciju *Interface*, a naziv *Figure*. Ovaj interfejs ima jednu metodu, *draw()* (Slika 1).



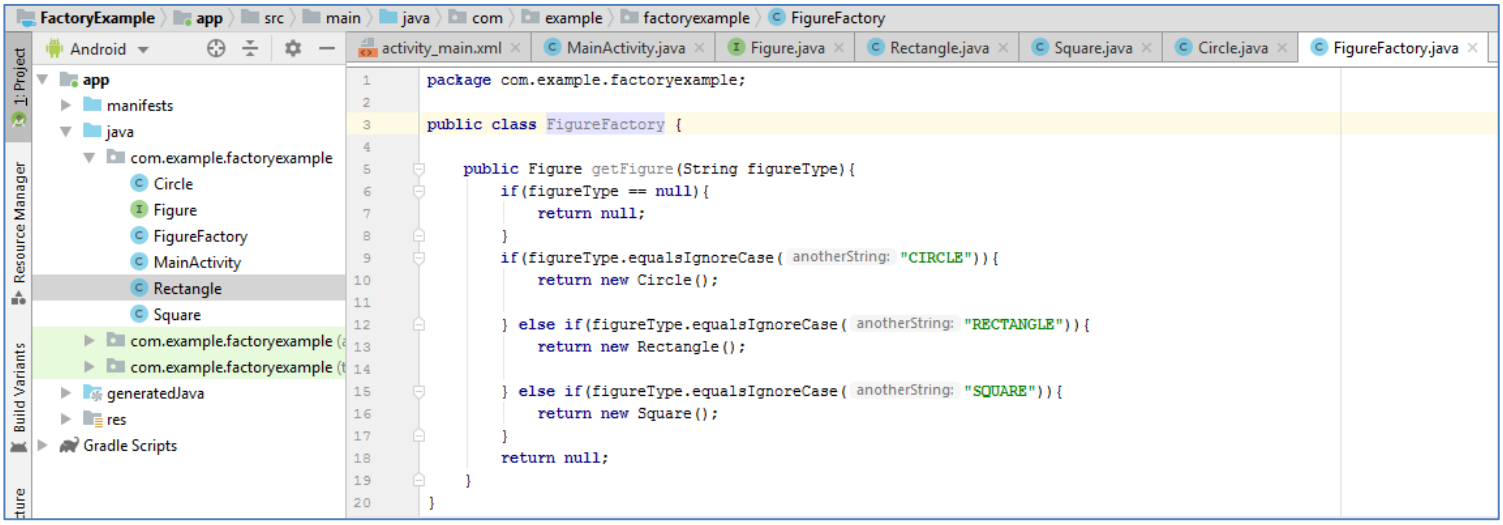
Slika 1 - Figure interfejs

- Kreiramo odgovarajuće klase koje će implementirati ovaj interfejs: Circle.java, Rectangle.java i Square.java (Slika 2).



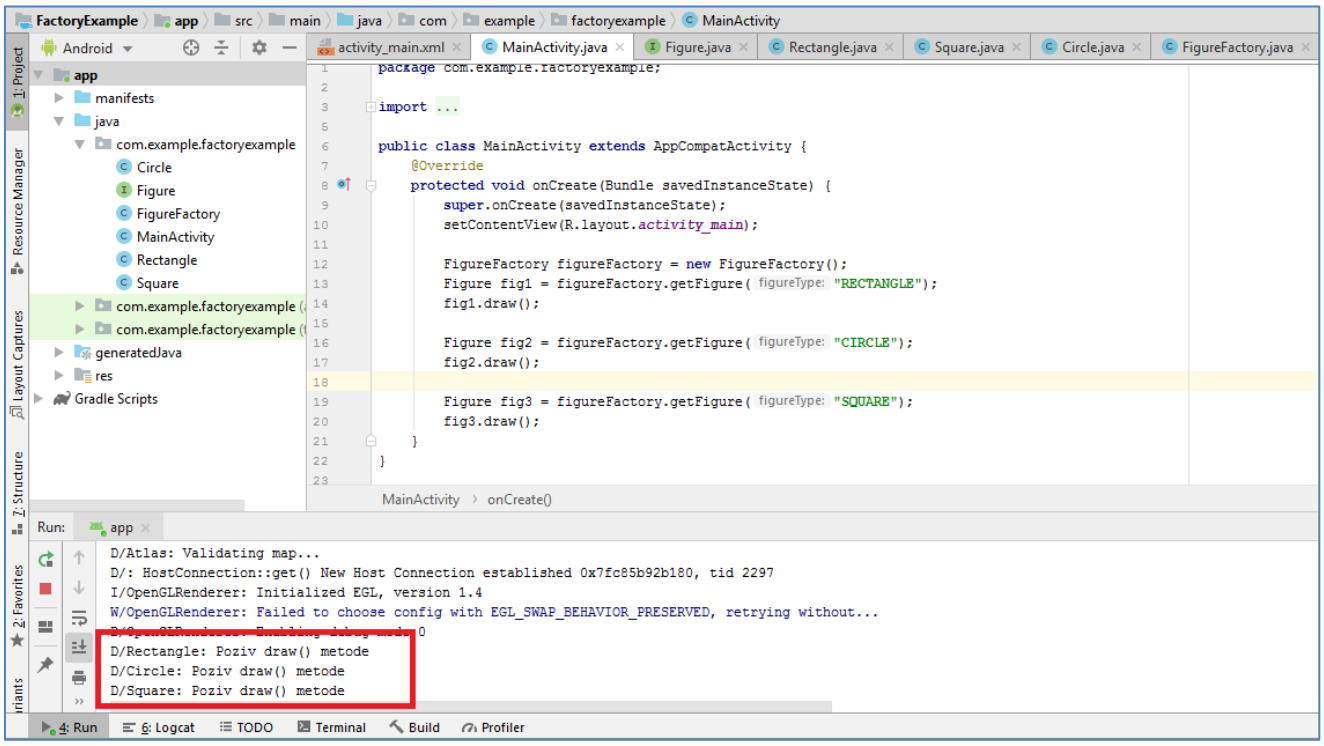
Slika 2 - Circle.java klasa

- Kreiramo i *FigureFactory*.java klasu koju ćemo koristiti za kreiranje različitih figura (Slika 3).



Slika 3 - FigureFactory.java klasa

- U okviru MainActivity – ja sada možemo pomoću Factory klase da pribavimo odgovarajući objekat. Nakon prevođenja i pokretanja projekta, posmatramo ispis i vidimo da su pozivane metode *draw()* iz odgovarajućih klasa. (Slika 4).



Slika 4 - MainActivity.java

Ovaj šablon koristimo kada:

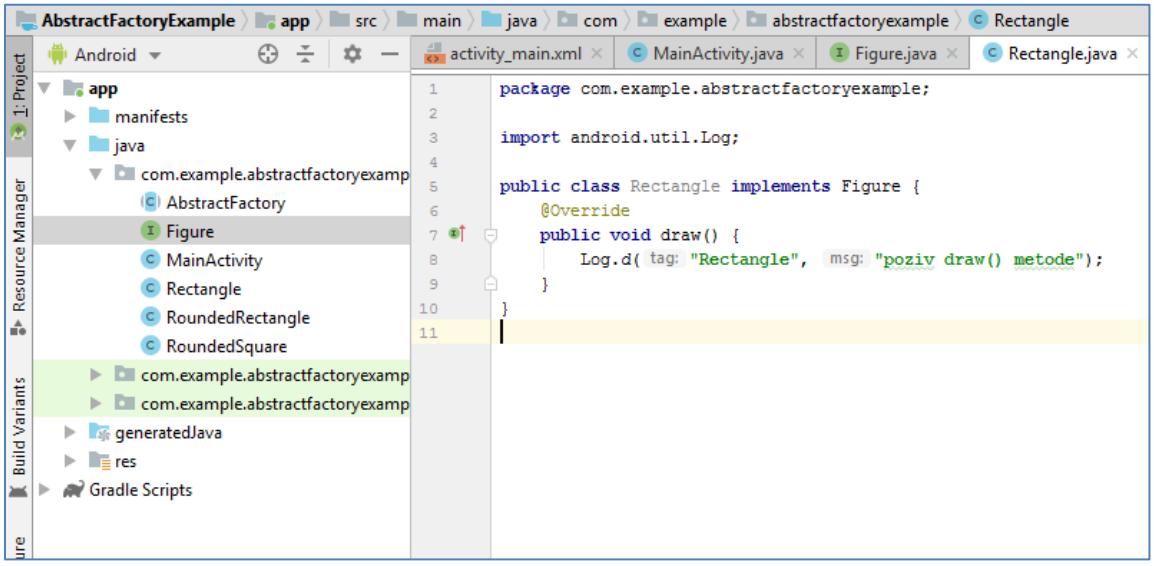
- Kreiranje objekata zavisi od datih ulaznih podataka.
- Kada se instanciranje klasa naslednica može izvršiti na više različitih načina ili želimo da ostavimo mogućnost da u budućnosti postoji više načina za kreiranje objekata.
- Kada je postupak kreiranja objekta malo kompleksniji. Kako bismo obezbedili da su ti koraci kreiranja centralizovani, koristimo ovaj šablon.

Apstraktni Factory šablon (engl. *Abstract Factory Pattern*)

Apstraktni Factory šablon se takođe naziva i "fabrika fabrika". Spada takođe pod tip šablona za kreiranje objekata. U ovom šablonu, interfejs je odgovoran za kreiranje jedne "fabrike" povezanih objekata, bez navođenja njihovih klasa.

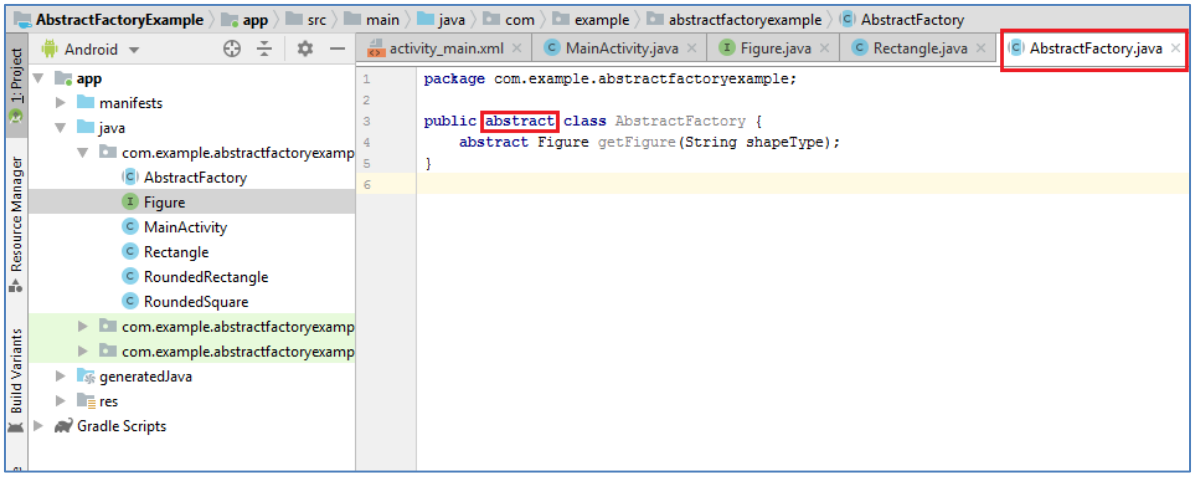
Primer

- Kreiramo novi projekat, AbstractFactoryExample.
- Kreiraćemo interfejs *Figura* kao i u prethodnom primeru, kao i četiri java klase koje ga implementiraju: *Rectangle.java*, *Square.java*, *RoundedRectangle.java* i *RoundedSquare.java*. Ove klase u implementaciji *draw()* metode treba da sadrže ispis koji nas obaveštava odakle je obavljen poziv (iz koje klase). Uraditi ovo po uzoru na prethodni primer takođe. Na slici (Slika 5) je data implementacija *Rectangle.java* klase, na isti način samostalno odraditi i preostale tri.



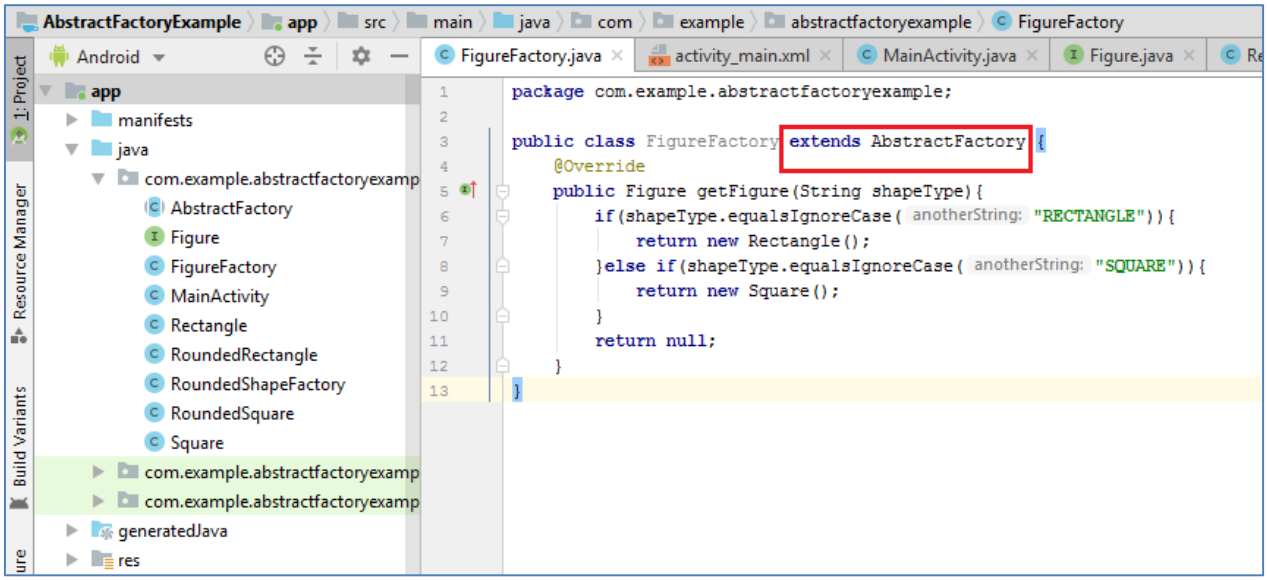
Slika 5 - *Rectangle.java* klasa koja implementira Figure interfejs

- Kreiramo apstraktnu klasu za pribavljanje "fabrika" različitih tipova figura, *AbstractFactory.java*. Ova klasa sadži samo jednu metodu, *getFigure()*, a tip povratne vrednosti je *Figure*.

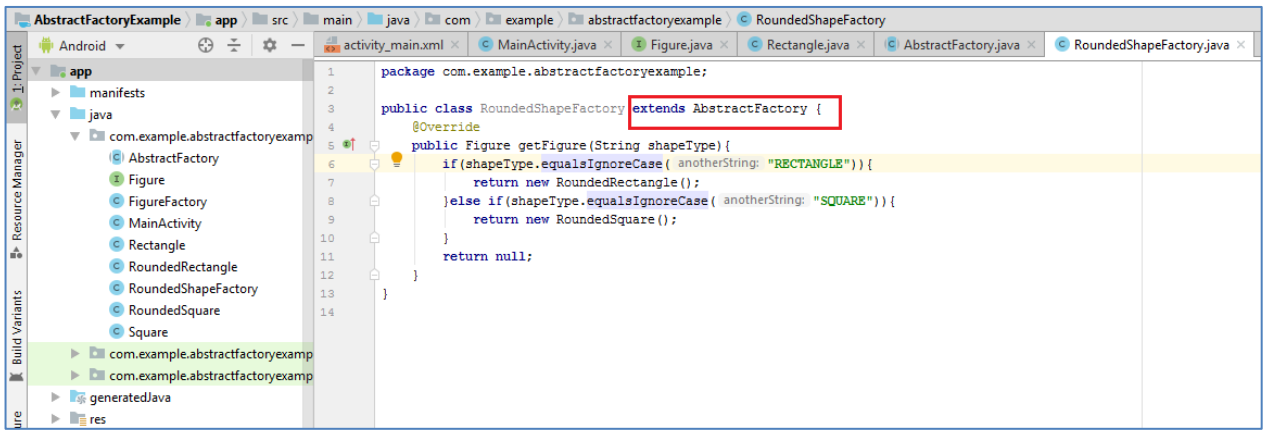


Slika 6 - Apstraktna klasa *AbstractFactory.java*

- Kreiramo klase koje nasleđuju AbstractFactory klasu i kreiraju objekte na osnovu prosleđenih vrednosti(Slika 7 i Slika 8).

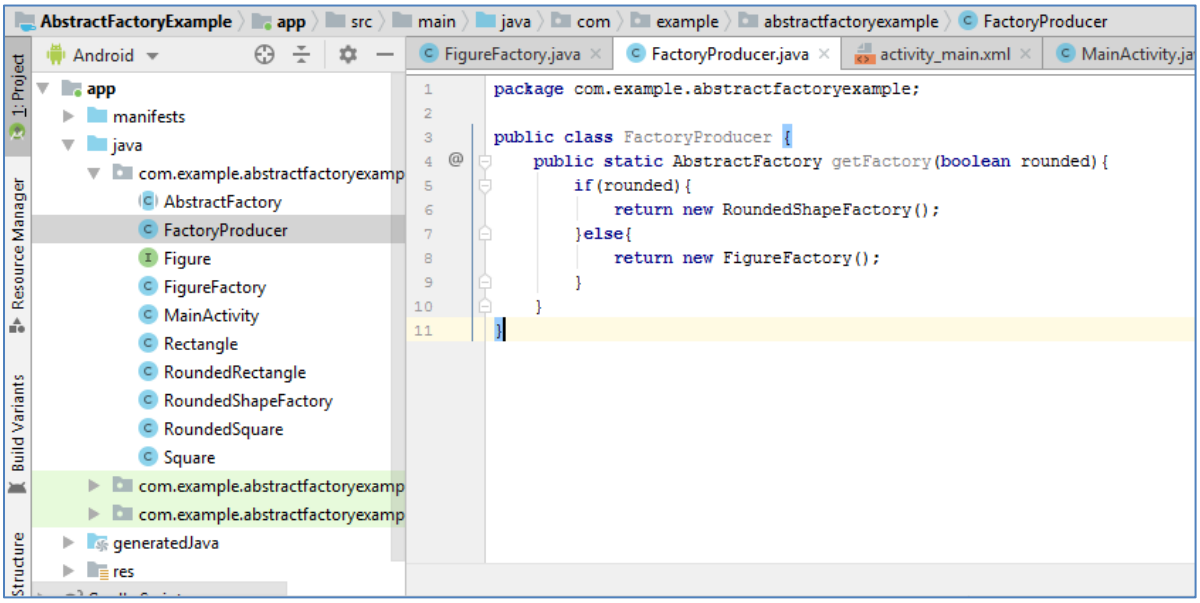


Slika 7 - *FigureFactory.java* klasa



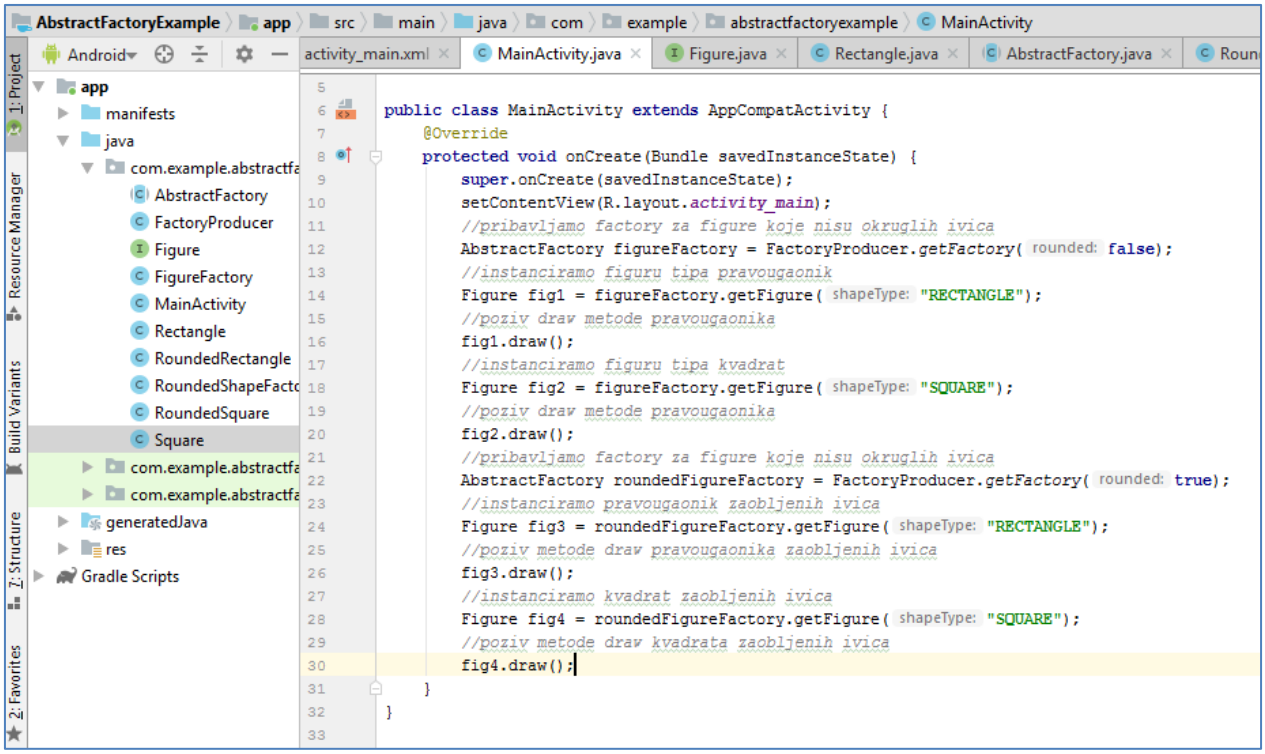
Slika 8 - *RoundedShapeFactory.java* klasa

- Sledeći korak nam je da kreiramo klasu koja generiše "fabrike", tj. proizvođače, na osnovu prosleđenih parametara (Slika 9).



Slika 9 - *FactoryProducer.java* klasa

- Sada možemo da iskoristimo *FactoryProducer* klasu da pribavimo *AbstractFactory* instancu, kako bismo preuzeli factory željene klase. Ovo ćemo opet demonstrirati u okviru *MainActivity*-ja.



Slika 10 - MainActivity

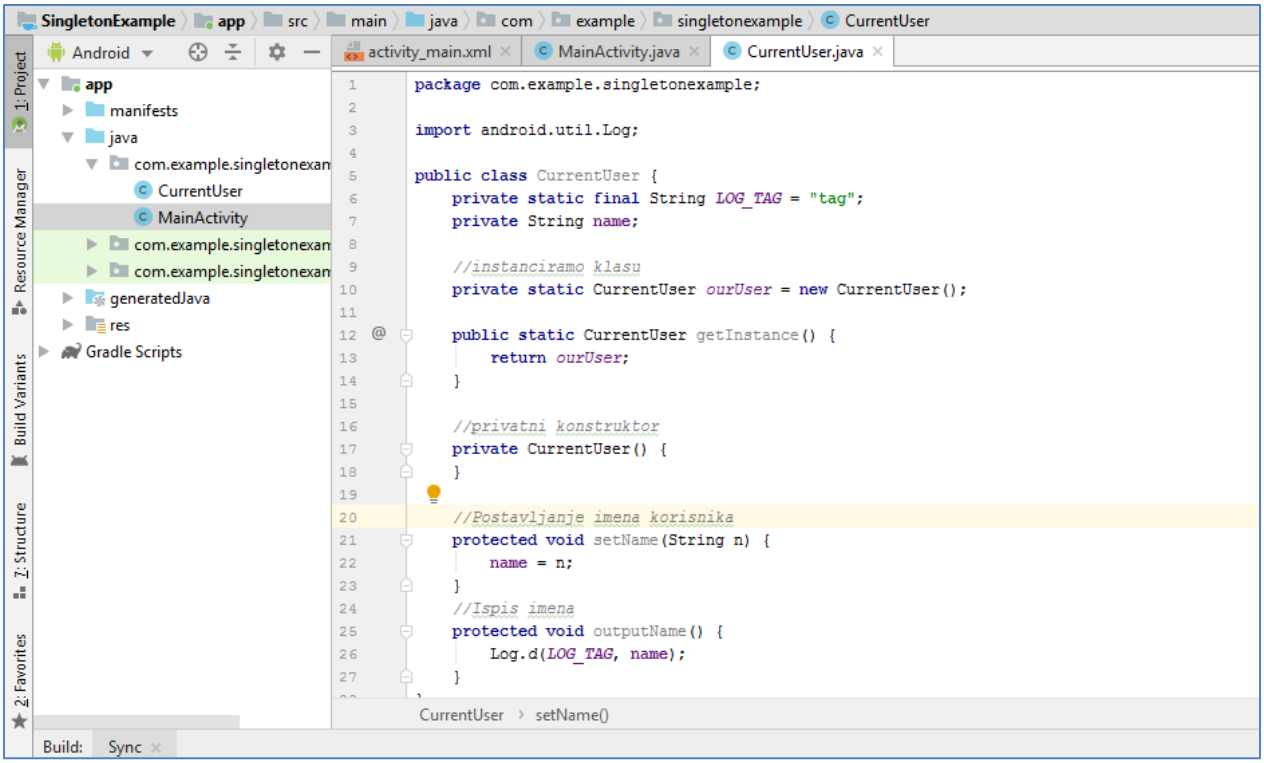
Singleton šablon (engl. *Singleton pattern*)

Singleton šablon takođe spada pod tip šablona za kreiranje objekata i jedan je od osnovnih i jednostavnijih šablona. Korišćenje ovog šablona podrazumeva postojanje jedne klase koja je zadužena za kreiranje objekta, vodeći računa o tome da se instancira samo jedan objekat. Iako se efekat može postići i korišćenjem *static* klase, singleton šablon treba koristiti kada želimo kompletan objekat, ali samo jedan, da bude dostupan bilo gde u okviru programa (npr. Želimo da predstavimo jednog korisnika koji je ulogovan na aplikaciju). Static klasu koristimo kada želimo da izvršimo neku metodu nad promenljivom koju joj i prosleđujemo.

Upotrebu ovog šablona prikazaćemo na primeru.

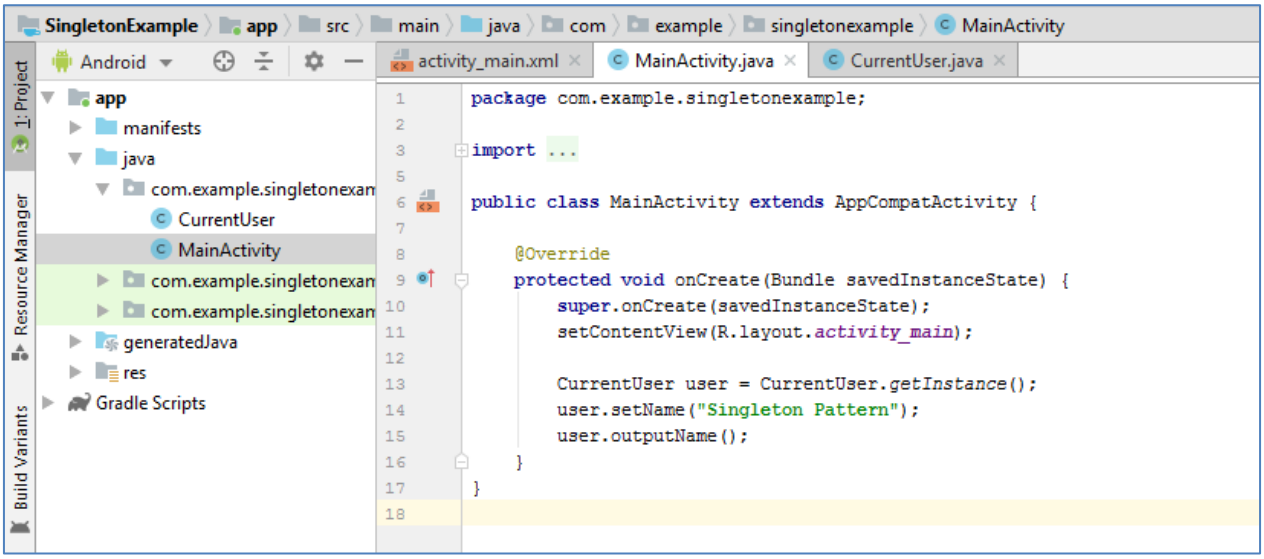
Primer

- Kreiramo novi Android Studio projekat *SingletonExample*, pa pod podmenijem **New** izaberemo **Singleton**. Ova klasa će nam predstavljati trenutno ulogovanog korisnika (uzimajući u obzir da trenutno može biti samo jedan korisnik ulogovan u aplikaciju). Ono što nam je u ovoj klasi potrebno jeste privatna instanca klase, privatni konstruktor (**Pitanje**: Koje ograničenje ovo postavlja?), kao i *get* metoda. Pored toga, kreiramo i metode za postavljanje atributa ***name*** korisnika, kao i metoda ispisa (Slika 11).



Slika 11 - Singleton klasa

- Šablon testiramo u okviru MainActivity – ja pozivom *setName*() metode, te ispisom.



Slika 12 - *MainActivity*

Ovaj šablon koristimo npr. kada rukujemo nekim asinhronim zadacima i želimo atributima da pristupamo bilo gde iz koda.