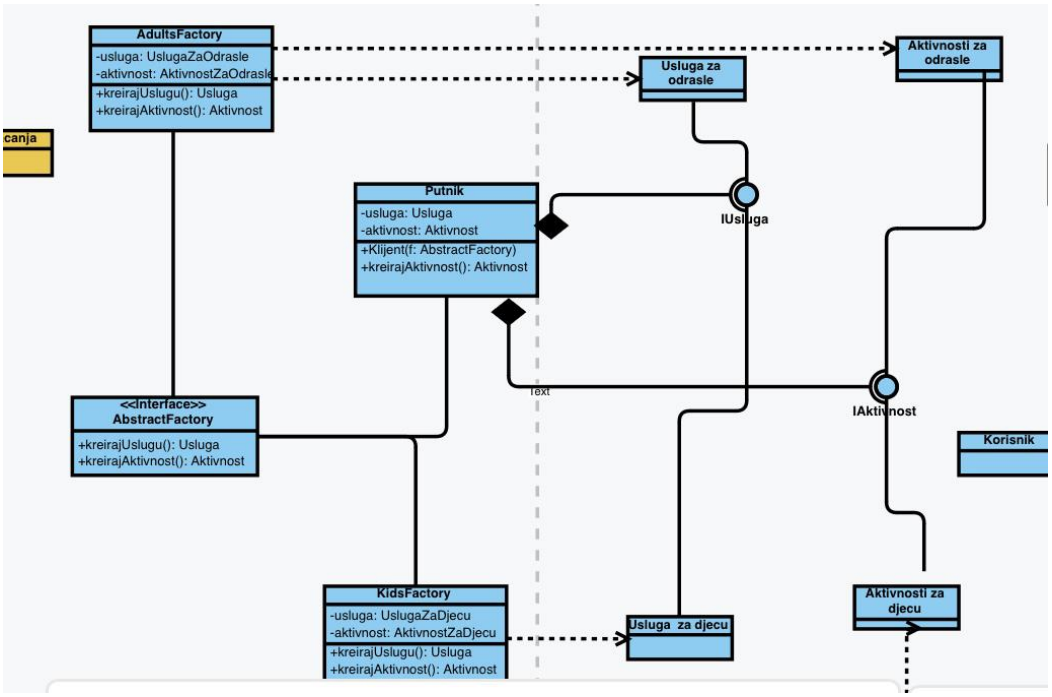
KREACIJSKI PATERNI

- **Singleton patern:** njegova uloga je osiguravanje da se klasa može insticirati samo jednom i da osigura globalni pristup(sve klase mogu pristupiti) kreiranoj instanci klase. Unutar klase je private static varijabla koja čuva jednu instancu klase i static metoda preko koje se pristupa Singleton klasi. Ovaj patern nismo iskoristili ali bi ga mogli primijeniti prilikom logiranja korisnika; potrebna nam je jedna instanca dostupna svim klijentima i imali bi jedinatveno upravljanje bazom u tom slucaju. Implementira se na nacin da imamo privatni staticki atribut gdje bi drzali intancu u klasi Prijava, privatni konstruktor kojem bi mogla pristupiti samo ta (Singleton) klasa i public metoda getIntance koja bi omogucavala pristup instanci klase, koja poziva prvi put konstruktor ukoliko instanca nije kreirana a svaki naredni put vraca instancu.

- **Abstract factory patern:** omogućava da se kreiraju familije povezanih objekata. Ukoliko postoji više tipova te različite klase koriste različite tipove, te klase postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specifiranjem pojedinačnih objekata. Korištenjem naslijeđivanja ukida se potreba za postojanjem if else uslova jer određeni tip fabrike sadrži određene tipove objekata i zna se koju klasu će instancirati. Ovaj patern smo iskoristili u sistemu. Cijena usluga i aktivnosti na letu zavise od dobi putnika te smo aktivnosti i usluge podijeli na adults(odrasle) i djecu. Akrivnosti bi mogle npr biti da su za djecu dostupni crtani filmovi za gledanje u toku leta dok bi odrasli imali obicne filmove, usluge mogu biti npr da odrasli mogu dobiti alkoholna pica u toku leta, djeca cokoladno mlijeko itd. Dodali smo dva nova interfejsa i cetiri nove klase. Interfejs Usluga implementira klase UslugaZaDjecu i UslugaZaOdrasle, interfejs Aktovnost impelementira klase AktivnostZaDjecu i AktivnostZaOdrasle. Na osnovu uzrasta putnika kreirane su fabrike KidsFactory i AdultsFactory. One objedinjuju proizvode UslugaZaDjecu i AktivnostZaDjecu tj UslugaZaOdrasle i AktivnostZaOdrasle.



- **Factory method patern:** Koristi se za instanciranje mnogo različitih podklasa uz pomoć factory metode, a samo instanciranje se radi kroz definisanje factory metode. Omogućava kreiranje objekata na način da klase odluče koju klasu instancirati. Factory method instancira odgovarajuću podklasu (izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja. Factory method pattern ispunjava Open-Closed Prinicple SOLID principa. Dakle on podržava otvorenost za proširivanje, a istovremeno je zatvoren za promjene. Npr. ukoliko želimo dodati neku novu vrstu projekcije, tada moramo dodati i novu podklasu koja će u sebi implementirati factory metodu samo za tu vrstu projekcije Ovaj patern nismo primijenili u nasem sistemu ali bi ga mogli iskoristiti kod kreiranja korisnika. Dodali bi interfejs Ikorisnik koji bi imao metodu dajUsera(), dodatno bi imali i Kreator klasu koja bi imala metodu FactoryMethod(). Ova metoda bi odlucivala koja ce se klasa instancirati.

- **Builder patern:** uloga builder paterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijaki proces može kreirati različite reprezentacije. U našem sistemu nismo koristili ovaj patern. Mogli bi ga primijeniti da smo imali funkcionalnaot da putnik, u zavisnosti od duljine leta, moze birati da li zeli da mu budu posluzeni dorucak, rucak i vecera. Preko ovog paterna mogli bi se napraviti obroci od nekih namirnica(dijelova) te bi imali klasu Kuhar koja bi implementirala inerefejs Ibuilder te klasu Sef koja bi omogucila da se konstruira objekar iz vise dijelova i sadrzavala bi metodu Construct. Dijelovi bi bile namirnice a sam produkt taj obrok . Imali bi klase namirnica npr Mlijeko, Sol itd. I klasu Objekt koja bi bila sastavljena od tih dijelova tj namirnica.

- **Prototype patern:** omogućava nam da kopiramo postojuće objekte bez da ovisimo o njihovim klasama. Tako smanjujemo kompleksnost kreiranja novog objekta. Ovaj patern smo implementirali u sistem na nacin da klasa Putnik zahtijeva kloniranje postojuceg objekta preko ineterfejsa IPrototype kojim se omogucava kreiranje postojucih objekata. Mi smo ovaj patern implementirali za Recenzije ocjene letova jer cesto putnici ostave iste ocjene, te bi se klonirale postojuce recenzije preko IPrototype interfejsa na zahtjev klase Let. Ovim smo izbjegli ponovno instanciranje klase, kloniranjem su prepisane sve vrijednosti klase Recenzija te se samo moze promijeniti vrijednost.