1. SP13: Aspektno orijentirano programiranje

1.1. Moze li se aspekt atribut definirati i ispred metode, a ne samo ispred klase?

1.2. Kada je potrebno koristiti cjelokupni AspectJ, odnosno kada nam Spring AOP nije dovoljan?

1.3. Koje su prednosti uporabe AOP naspram obrasca dekoratora?

1.4. Što je aspektno-orijentirano programiranje? Navedi barem tri dostupne implementacije aspektno-orijentiranog programiranja (bilo kao jezik sam po sebi ili kao vanjska biblioteka).

1.5. Zašto bi trebao koristiti AOP prije dependency injection-a? Korištenje dependency injectiona se ipak drži glavnih OO principa.

1.6. Može li se na neki način aspekt definirati unutar tijela neke metode?

1.7. Povećavamo li korištenjem aspektne paradigme međuovisnost komponenata/klasa? Koje načelo dizaniranja dobrog koda možemo izravno povezati sa aspektnom? (Spearation of Concerns)

1.8. Nabroji 3 primjera gdje se često koristi aspektno orijentirano programiranje i razloge zašto.

1.9. Mogu li aspekti savjetovati druge aspekte?

2. KP4: LINQ

2.1. Npr ako imamo:public static void Disp(this T[] arr){System.out.print(arr.Length*2); } public static void Disp(this String[] arr){System.out.print(arr.Length*3);} I u mainu string[] arr=new string[]{"abc"}; arr.Disp(); Koji Disp se poziva?

2.2. Kad imamo reference na dvije anonimne klase istog prototipa, je li moguće postaviti da jedna referenca pokazuje na drugu, te potom baratati njezinim atributima? Ili je svaka referenca na bilo koju anonimnu klasu sama za sebe specifična?

2.3. Postoji li neka proširena metoda kojom se može zaobići/preskočiti prvih ili zadnjih n rezultata u upitu?

2.4. Ako postoji kolekcija object-a, mogu li se u nju dodavati anonimni tipovi? Zašto (ne)?

2.5. Koji je razlog korištenja LINQ upita naredbama (in, from, select...) naspram metoda proširenja (.Select(), .Where() ...) ?

2.6. Koja je prednost koristenja LINQ nad pohranjenim procedurama?

2.7. Dohvaća li query varijabla podatke odmah pri stvaranju varijable ili u nekom drugom trenutku?

2.8. Je li moguće izvršiti sljedeći dio koda te ako nije, što treba promijeniti? var customers = new[]{ new { Name = "Marco", Discount = 2.5 }, new { Name = "Paolo", Discount = 3.0 } }; IEnumerable(string) customerNames = from customers as c where c.Discount > 3 select new { c.Name };//napomena: ()

2.9. Na koji način metoda proširenja, koja kao argument prima polje podataka, može postati univerzalna za sve tipove polja?

2.10. Kako se u LINQ-u postize left outer join, a kako inner join?

2.11. U C# projektu postoji klasa s konstruktorom "public AB(int a, float b)". U metodi tog projekta postoji polje "var polje = new[] {new { A = 1, B = 1.0f, C = "1" }, ...};". Nadopuni kod "IEnumerable rezultat = ___;" LINQ upitom tako da u variabli rezultat budu samo elementi kojima je C jednak "1".

3. SP15: Funkcijsko programiranje u F#

3.1. Koje su prednosti funkcijskog programiranja u odnosu na objektno orijetirano programiranje?

3.2. Budući da je F# primarno funkcijski jezik, postoje li u njemu uopće varijable ili su one zapravo funkcije koje ne primaju nijedan argument i kao povratnu vrijednost vraćaju onaj iznos koji je konceptualno dodijeljen toj "varijabli"?

3.3. Što je to "lazy computation" u F#?

3.4. Napišite kod u f# koji stvara listu brojeva, funkciju koja provjerava proste brojeve u listi, i liniju koda koja filtrira listu po toj funkciji i zatim iterira ispis članova

3.5. Što je 'self' u F#-u?
Self identifikator je referenca na trenutni objekt. Ekvivalentan je ključnoj riječi “this” u Javi i C#. Možemo ga nazvati kako god želimo u F#.

3.6. Što je to čista funkcija (pure function)?

3.7. Stvorimo varijablu "let myInt = 5". I F# prepozna da je to integer. I onda od te varijable oduzmemo 0.01. "myInt = myInt - 0.01". Hoće li tada F#

promijeniti svoje mišljenje iz integera u real i izvesti operaciju ili će izbaciti error?

3.8. Pošto funkciju s parametrima u F# zovemo nazivom funkcije te zatim parametre navodimo razmacima, kako bi predali kao parametar funkciju s parametrima?

3.9. Koja je razlika između tupplea, listi i polja u F#?

3.10. Zašto je nepromjenjivost jedno od važnijih svojstava u funkcijskom programiranju?

3.11. Koja je razlika između List i Sequence? Koje je bolje koristiti i kada?

3.12. Čemu služi "unit" tip u programskom jeziku F#?

3.13. Za naredbe u F#-u: let neg x = x *(-1) let square x = x*x, Što se dobije ispisivanjem neg >> square ?

4. SP16: Stat i din upravljanje memorijom 1

4.1. Što je krivo u navedenom isječku koda? int main() { int *pointer = (int *)malloc(sizeof(int)); pointer = NULL; free(pointer); }

4.2. Na koji način operacijski sustav upravlja sa "rupama" na heap-u? Na primjer, ako nema dovoljno velike rupe u memoriji, koju je korisnik zatražio, hoće li ono realokacijom ostalih objekata napraviti dovoljno veliku rupu ili će odbiti alokaciju?

4.3. Postoje li mehanizmi za ručno oslobadjanje memorije u jezicima koji imaju ugrađen Garbage Collector? Ako da, navedi primjer kada je dobro taj pristup koristiti.

4.4. Može li se garbage collector ručno pozivati u jezicima koji ga koriste?

4.5. Navedi primjer u kojem je bolje statičko upravljanje memorijom od dinamičkog i primjer u kojem je dinamičko upravljanje memorijom bolje nego statičko.

4.6. Što se dešava kada oslobodimo blokove memorije različite veličine razlčitim redosljedom?

4.7. Na koji način je moguće u C++ rješiti problem curenja memorije? Na koji način je potrebno modificirati normalne pokazivace kako bi to ostvarili? (Smart pointers)

4.8. Postoji li alternativni nacin oslobadanja memorije osim operatorom delete?

4.9. double* p = new double{11.}; p = new double{12.}; delete p; - Kako se naziva problem koji nastaje u ovom odsječku? Koji je ispravan način za brisanje pokazivača?

4.10. Koja je razlika između spremanja objekta na stog i na gomilu? Jesu li veličine spremljenog objekta jednake? Zašto?

5. SP17: Stat i din upravljanje memorijom 2

5.1. Na koji način substring metoda u Javi može uzrokovati curenje memorije?

5.2. Kada objekt postaje podložan garbage collectionu? Tj, kako garbage collector zna koje objekte smije "odbaciti" ?

5.3. Koje dvije vrste fragmentacije postoje i koje su njihove razlike?

5.4. Što je to fragmentacija i kako dolazi do nje?

5.5. int i; int main() { int j; int *k = (int *) malloc (sizeof(int)); } Gdje su u memoriji pohranjene varijable i, j, k i zašto?

5.6. Napiši C++ odsječak koda koji će osloboditi memoriju zauzetu na sljedećem odsječku koda (uz predpostavku da imate pristup variabli "A* a"): "a = new A(); a->b = new B(); a->b->c = new C();"

5.7. Kada C# pokreće garbage collection?

5.8. U javi naredbom System.gc() mozemo dati hint JVM da pokrene garbage collector koji se ne mora odmah pokrenuti. Zasto nemamo punu kontrolu nad garbage collectorom?

5.9. Ako klasa napisana u c++ sadrži varijable int a i float b, gdje će biti alocirana memorija i koliko B će iznositi ako smo objekt stvorili sa ključnom riječi new ?