Mündəricaat

(Bunlar sual deyil cavab vermeyin sadece hansi movzulari ehate edir onlar qeyd olunub. Suallar bundan sonra aşağıdan başlayır):

1. Design Patterns (1-9)
2. Date (10-11)
3. Enum (12-17)
4. Fayllar (18-28)
5. Annotation (29-30)
6. Reflection(31)
7. Debug Rejimi (32)
8. Ant, Maven, Gradle (Ant+Maven) (33)
9. Lombok, Apache Commons (34-35)
10. Git (36-38)
11. TCP (39-43)
12. Threadlər (44-56)

Suallar (Bunlar isə suallardır və cavablandırılmalıdırlar. Rəqəmləri və sualları silməyin, sualların rəngini, şriftini dəyişməyin, adəcə aşağı düşüb cavabınızı yazın):

1. Design Pattern dedikdə nə başa düşürsünüz?

Cavab: Design Patterns proqram inkişaf etdiriciləri tərəfindən üzləşdikləri əsas problemlərin həlləridir(uzun müddət ərzində əmələ gəmişdir). Design Patterns yazıçıları adətən GOF adlandırılır.

1. Singleton Design Pattern nədir?

Cavab: Singleton Design Pattern hər hansı bir class-ın birdən çox obyektinin yaradılmasının qarşısını alır. Class-ın obyekti bir-başa yaradılmasın deye onun konstruktorunu private edirik.

1. Proxy Pattern nədir?

Cavab: Proxy Pattern-də arada bir class dayanır və əsas class bu class ilə işləyir. Proxy Pattern əsas class-a hansı class-larla işlədiyini bildirmir.

1. Builder Pattern nədir?

Cavab: Builder Pattern class-ın obyekti yaradılan zaman çox sayda konstruktor çağırmalı oluruq, bunun qarşısını almaq üçün istifadə olunur.

1. FactoryPattern nədir və abstraction və interface-in tətbiqi

Cavab: FactoryPattern-də biz Factory class-ı ilə işləyirik. Məsələn 3 class var dordbucaq, üçbucaq, kvadrat. Bu class-lar da fiqurlar classın-dan implements edib. Factory classında əsas(demo) classından gələcək sözə uyğun Məsələn: üçbucaq deməli, üçbucaq classının obyektini qaytaracaq. Bu üçü də fiqurlar classını implements etdiyinə görə əsas(Demo) classında hansı class ilə işlədiyimizi görməyəcəyik. Biz yalnız Factory classının obyektin yaradib, metodunu çağıracağıq və geriyə gələn classı da polymorphismə gorə fiqurlar classına bərabər edəcəyik. Nəticədə biz Factory vasitəsilə digər classlar ilə işlədik, lakin bilmədik hansı ilı işlədik. Və üçbucaq, dördbucaq, kvadrat classında nəsə dəyişsə Factory classında dəyişməyimiz bəs edəcəkdir.

1. AbstractFactoryPattern nədir

Cavab: Color və Shape interfaceimiz var. Və bunlara uyğun red, green, black və rectangle, square , circle class-larımız var hansı ki, Color, Shape-i implements ediblər. ShapeFactory, ColorFactory classlar var ki, onlar da geriyə Shape və Color qəbul edirlər. AbstractFactory abstract class-dır. ShapeFactory,ColorFactory də extends edib AbstractFactory-dən. FactoryProducer və AbstractFactory classlar var. Və burada da AbstractFactoryProducer FactoryProducer ilə işləyir. Və arxada nə dəyişikliklər olacaq ona diskomfort yaratmayacaq. Heç bir problemə gətirib çıxartmayacaq. Bu Pattern-dən nə ki, digər şəxslərin proyekləri ilə işləyərkən, həmçinin öz proyektlərimizdə də istifadə etmək lazımdır.

1. SOLID principles nələrdən ibarətdir? 5 prinsipin hər birini yazın. Ay müəllim birini unutdum ee vaxtım da olmadı araşdırım demək lazım deyil. İnterview da belə cavab keçmir

Cavab: S- hər bir class bir işi yerinə yetirməlidir.

O-elə bir class yaratmaq lazımdır ki, o genişlənə bilsin.(extends, implement ilə)

L-Bu prinsipdə extends etdiyimiz classın həqiqətən çaşqınlığa gətirib çıxartmayacağına nəzarət edir.

İ-Müştərini məcbur etmək olmaz ki, ona lazım olmayan metodları implement eləsin.

D-konkretləşməyin, abstractlaşın.

1. Mutable class dedikdə nə başa düşürsünüz? Nümunə gətirin

Cavab: Mutable class classın obyekti yarandıqdan sonra dəyişənin dəyərini dəyişdirmək mümkündür.

Nümunə:

public static void main(String[] args) {

NewClass d=new NewClass("ss");

System.out.println(d.getName());

d.setName("dd");

System.out.println(d.getName());

}

private String name;

NewClass(String name){

this.name=name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

1. Immutable class dedikdə nə başa düşürsünüz? Nümunə gətirin

Cavab: Immutable class classın obyekti yarandıqdan sonra dəyişənin dəyərini dəyişdirmək mümkün deyil.

Nümunə: public static void main(String[] args) {

NewClass instance = instance("dd");

System.out.println(instance.getName());

}

private String name;

private static NewClass df=null;

private NewClass(String name){

this.name=name;

}

public static NewClass instance(String name){

if (df==null) {

return new NewClass(name);

}

return null;

}

public String getName() {

return name;

}

1. Date nədir?

Cavab: Date Java-da bir class-dır. Özündə indiki zamanı və tarixi saxlayır. Bu classın bir çox metodları var. Bu indiki tarixi, saatı, saniyəni və s. Metodları var.

1. SimpleDateFormatter nədir, parse və format nədir?

Cavab: SimpleDateFormatter bir classdır. String qəbul edir. Bura daxil edilən String hansı formadadısa Date classına da həmin formatda String göndərmək lazımdır. Və Ayı günü, saatı, ili göstərəcək.

Nümunə: SimpleDateFormat sdf=new SimpleDateFormat("dd:mm:YY HH:mm:ss");

String format = "12:12:18 19:22:33";

Date d=sdf.parse(format);

String f=sdf.format(d);

System.out.println(f);

System.out.println(d);

Burada parse daxil olan string-i SimpleDateFormatın qəbul etdiyi "dd:mm:YY HH:mm:ss" formata uyğun olaraq Date tipinə çevirir. Format SimpleDateFormatın qəbul etdiyi "dd:mm:YY HH:mm:ss" formata uyğun tarixi stringə çevirir.

1. Enum nədir? Bilmirsinizsə unutmusunuzsa açın oxuyun yarımçıq cavab verməyin.  
     
   http://javacigenclik.blogspot.com/2012/03/enum-ndir-n-ucun-istifad-olunur.html

Cavab: Enum sabit dəyişənlərlə işləmək üçün istifadə olunur. Məsələn: Calculator api-də TOPLA,CİX,VUR,BOL dəyişənləri enum kimi çıxış edir. Enumun adın yazdıqda public enum Emeliyyat{ } yazırıq. İçəridə isə TOPLA,CİX,VUR,BOL kimi obyektlər olur hansı ki, Emeliyyat obyektindən extends ediblər.

1. Enum necə elan olunur?

Cavab: Bİrinci class-ın adı belə ifadə olunur. public enum Emeliyyat{ } .İçəridə isə TOPLA,CİX,VUR,BOL kimi obyektlər olur hansı ki, Emeliyyat obyektindən extends ediblər. Topla('+'),

Cix('-'),

Vur('\*'),

Bol('/');

Bu obyektlərin mötərizələrinin sonunda vergül, sondakında isə nöqqtə-vergül qoyulur. İşə salmaq üçün isə Emeliyyat to = Emeliyyat.Topla; yazırıq.

1. Enum classının içindəki values(), ordinal(), valueOf() methodları nə iş görür

Cavab: Enum classının içindəki values() classın içindəki obyektləri qaytarır.

valueOf(“”) isə içəriyə gödərdiyin Stringə uyğun obyekti qaytarır.

ordinal()-Bu obyektin sırasını qaytarır.

1. Enum-da constructor və dəyişənlərin necə elan edə bilərik nümunə gətirin

Cavab: public enum Emeliyyat {

Topla('+'),

Cix('-'),

Vur('\*'),

Bol('/');

private char c;

Emeliyyat(char g) {

this.c = g;

}

public double sss(double a, double b) {

if (c == '+') {

return a + b;

} else if (c == '-') {

return a - b;

} else if (c == '\*') {

return a \* b;

} else if (c == '/') {

return a / b;

}

return 0;

}

}

1. Enum-da methodları necə elan edirik, nümunə gətirin

Cavab: public enum Emeliyyat implements Heasbla {

Topla {

@Override

public double emeliyyat(double f, double g) {

return f + g;

}

},

Vur {

@Override

public double emeliyyat(double f, double g) {

return f \* g;

}

}

}

public interface Hesabla {

double emeliyyat(double f,double g);

}

1. Enum elan edəndə arxada hansı class yaranır, bu class nəyi extend edir? Bilmirsinizsə oxuyun cavab verin:  
   http://javacigenclik.blogspot.com/2016/07/slind-enum-ndir.html

Cavab: Enum elan edəndə Məsələn: Emeliyyat enumu yaradanda Emeliyyat classı yaranır. İçində də topla,cix, vur kimi dəyişənli obyektlər yaranır. Lakin bu class özümüz yaratdıqda Enum classından extends edə bilmirik.Çünki Enum classı finaldır. Lakin özümüz yaratmadıqda java vasitəsilə yaratdıqda default olaraq ona icazə verir extends etməyə(bundan Enum<Emeliyyat>

1. Məlumatın File-a IO ilə yazılması və oxunması necə olur? methodu yazın və methodda yazılan kodları izah edin nə nə üçün yazılıb

Cavab: public static void writeBytes(String fileName, byte[] data) throws Exception {

FileOutputStream fop = new FileOutputStream(fileName);**// Fayli Out colden iceriye kanal acir**

// get the content in bytes

fop.write(data);**//bu kanalin deyiseni ile byte-lari kanalin yoneldiyi directoryaya yazir**

fop.flush();**//tesdiqleyir yazini**

fop.close();//kanali kill edir

System.out.println("Done");

}

public static byte[] readBytes(String fileName) throws Exception {

File file = new File(fileName);**//faylin direktoriyasina uygun File obyekti duzelir**

try (FileInputStream fileInputStream = new FileInputStream(file);) {**// bu iceriden cole kanal acir File obyektine gore**

byte[] bytesArray = new byte[(int) file.length()];**//melumatin uzunluguna gore array yaradir**

//read file into bytes[]

fileInputStream.read(bytesArray);**//kanalin read metodu ile melumati oxuyur bayt arraya doldurur**

return bytesArray;

}

}

1. Məlumatın File-a NIO ilə yazılması və oxunması necə olur? methodu yazın və methodda yazılan kodları izah edin nə nə üçün yazılıb

Cavab: public static void writeBytes(byte[] data, String fileName) throws Exception {

Path filePath = Paths.get(fileName);**//gelen Stringe uygun Path(cigir) obyekti yaradir**

Files.write(filePath, data);**//Files obyekti vasitesile byte-ile filePath directory-sine yazir**

}

public static byte[] readBytesNio(String fileName) throws Exception {

Path filePath = Paths.get(fileName);**//gelen Stringe uygun Path(cigir) obyekti yaradir**

byte[] byteArray = Files.readAllBytes(filePath);**//melumatin yerlesdiyi directory-den Files obyekti vasitesile byte-lari oxuyur massive doldurur**

return byteArray;

}

1. FileWriter ilə BufferedWriter arasındakı fərq nədir?

Cavab: BufferedWriter daha effektivdir. Birdən çox faylı yaza və bağlaya(close) bilərik. FileWriter hər yazma üçün əməliyyat sisteminə müraciət edir. Ona görə ondan az, amma içində məlumat çox olan hallar üçün istifadə etmək daha məqsədəuyğundur.

1. Marker interface nədir?

Cavab: Marker interface- String obyekti Serializable-dan implements edib. Bu ona xüsusiyyət verir. BU xüsusiyyət verməsi onun is a Serializable olduğunu göstərmir. Bundan istifadə edən bilir ki, belə xüsusiyyəti var.

1. Serialization interface-ini nə üçün istifadə edirik?

Cavab: Serialization interface-i bundan implements edən class-a xüsusiyyət verir ki, fayla yazıla bilər.

1. Obyekti fayla necə yazırlar və necə oxuyurlar?

Cavab: public static Object readFileDeserialize(String name) throws Exception {

Object obj = null;

FileInputStream fi = new FileInputStream(name);

try (ObjectInputStream in = new ObjectInputStream(fi)) {

obj = in.readObject();

} finally {

return obj;

}

}

public static void writeObjectToFile(Serializable object, String name) throws Exception {

try (FileOutputStream fout = new FileOutputStream(name);

ObjectOutputStream oos = new ObjectOutputStream(fout);) {

oos.writeObject(object);

}

}

1. Transient keyword-ü nə üçün istifadə olunur?

Cavab: Transient keyvordu obyektin içində yazdığımız dəyişəni fayla yazılmasını istəmiriksə, transient-dən istifadə edirik.

1. serialVersionUID nə üçün istifadə olunur?

Cavab: serialVersionUIDfayla obyekti yazandan sonra obyektə yeni dəyişən əlavə edib, yazmağı siləndən sonra oxuduqda exception alırıq. Bu exception almamaq üçün serialVersionUID-dən istifadə olunur.

Private static final long serialVersionUİD=1L;

1. File-a yazma və ya oxuma zamanı sonda hansı obyekt close olunur və bu obyekti try-with-resources ilə automatic necə close edə bilərsiniz?

Cavab: (File,Object) (Input,Output)Stream close olunur/ try(BU obyekti bu mötərizə daxilinə yazırıq){ } sonra istəsək catch(Exception ex) ya da finally yazmağımız bəs edər.

1. Obyekti try-with-resources ilə close edə bilmək üçün class hansı interface-i implements etməlidir

Cavab: Obyekti try-with-resources ilə close edə bilmək üçün class Closeable interfae-ni implements etməlidir.

1. Bir obyekti fayla yaza bilmək üçün həmin class hansı interface-i implements etməlidir?

Cavab: Bir obyekti fayla yaza bilmək üçün həmin class Serializable interface-ni implements etməlidir.

1. Annotation nədir, öz annotationımızı necə yarada bilərik?

Cavab: Bir çox Annotation-lar vardır. Metodun classın dəyişənin üzərində @ ilə və adı yazılaraq elan olunur. Məsələn: @Override override-da səhv olduqda altında qırmızı xətt əmələ gəlir. Səhvə yol verdiyimizi bildirir.

Öz annotationumuz belə yaranır:

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

public @interface JsonField {

}

Burada class yerinə @interface keyvordundan istifadə olunur.

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

1. AnnotationProcessor nədir?

Cavab: AnnotationProcessor vasitəsilə annotation-ları compile zamanı analiz edə bilir. Və bu annotation-ları yerinə yetirir. Biz öz yaratdığımız annotationumuzu da annotation processora qoşa bilərik.

1. Reflection nədir, Reflection istifadə edərək Classın içindəki methodların adlarını necə çap edə bilərsiniz?

Cavab: Reflection metodları,classları,interface-ləri compile zamanı bilmədən, Runtime zamanı onları yoxlaya bilir.

Reflection istifadə edərək Classın içindəki methodların adlarını çap etmək:

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.METHOD)

public @interface MyAnnotation {

}

public class Reflection {

public static void main(String[] args) {

FileUtility rr=new FileUtility();

method(rr);

}

public static void method(Object obj) {

Method[] methods=obj.getClass().getDeclaredMethods();

for (int i = 0; i < methods.length; i++) {

Method d = methods[i];

Annotation[] f = d.getAnnotations();

String name = methods[i].getName();

System.out.println(name);

}

}

}

public class BMW {

public static class A {

}

@MyAnnotation

public void start() {

System.out.println("BMW driving");

}

@MyAnnotation

public void stop() {

}

@MyAnnotation

public void speedup() {

}

}

1. Debug rejimində işləmək dedikdə nə başa düşürsünüz və Netbeansdə bunu necə əldə edə bilərik.

Cavab: Debug rejimi vasitəsilə yazdığımız proyektdə harada Exception olduğunu Compile zamanı öyrənə(yoxlaya) bilirik. Beləliklə debug rejimi köməyilə bir-bir metodlar daxilinə girib exception-muzun harada olduğunun qarşısını almış oluruq. Debug rejimində Step over, Step into kimi əmrlər var. Step over vasitəsilə bir-bir sətirlər üzrə yoxlayırıq. Step into ilə metodlar daxilinə daxil oluruq . NetBeans-də Debug rejimini yerinə yetirmək üçün istədiyimiz sətrin sıra nömrəsinin üzərinə basırıq, yuxari hissədə run-nın ətrafında olan debug project əmrini seçirik.

1. Ant, Maven, Gradle (Ant+Maven) - Bu gördükləriniz nədir və bunlardan hər hansısa biri olmadan biz kodumuzu compile edə bilərikmi? Əgər edə bilərik deyirsinizsə o zaman bunlar niyə lazımdır?

Cavab: Ant, Maven, Gradle-i proyekt yaratmaq üçün seçirik. NetBeans-də adicə java application seçdikdə ant proyekt yaranır. Bunun əvəzinə Maven, Gradle da seçmək olardı. Ant, Maven fərqləri ondadır ki, ant-da lifecycle mövcuddur. Əvvəl nəyi etsin sonra nəyi etsin. Məsələn, ilk compile eləsin sonra dist faylını yaratsın. Maven-də isə yalnız dependencies mövcuddur . Bura jar-ları rahatlıqla daxil edirik. Bunu isə Maven ilk olaraq global-da hər hansı bir jar-ı axtarır, tapmadıqda local-da axtarır. Tapdıqda pom.xml-a əlavə edir. Gradle-da isə həm lifecycle, həm də dependencies var. Əlbəttə ki, bunlardan birinin olmamağı heç bir problem yaratmır. Bu(build tools)-lar proyekti yaradan zaman lazim olur və bunlar bir çox üstünluklər verir, Maven jar-ların rahatlıqla daxil edilməsini təşkil edir, ant lifecycle-a təsir etmək imkanı verməklə bir çox imkanlar təklif edirlər.

1. Lombok nədir bizə nə kimi köməyi dəyir?

Cavab: Lombok jar faylıdır. Və Lombok-un rəsmi saytından yükləyirik, ant-a özümüz daxil edirik, Maven-də isə jar faylın içindəki məsələn: hər hansı annotation-dan istifadə etdikdə kənarda balon şəkli çıxır və add dependenci əmrini seçirik, və lombok jar faylı daxil olur. Lombok daxilində bir çox metodlar, annotationlar mövcuddur. Bunun sayəsində məsələn: @SneakyThrows vasitəsilə exceptionları tutmağa ehtiyac olmur.

1. Apache Commons nədir bizə nə kimi köməyi dəyir?

Cavab: Apache Commons da Lombok kimi jar faylıdır. Proqramçıya bir çox imkanlar verir. Apache Commons-da əvvəlcədən mövcud olan bir çox class-lar, metodlar var. Məsələn StringUtils.isEmpty(), StringUtils.isBlank() kimi imkanlar var. Apache Commons-un Math, Collection, İO, Lang və s üçün Jar faylları var.

1. Git nedir, subversion nedir, github, bitbucket nedir?

Cavab: Git proyekt üzərində dəyişikliklərə nəzarət edilməsi, bir çox insanların qrup şəklində fayl üzərində işləməsi üçün nəzarət sistemidir. Digər bir nəzarət sistemi subversion-dur. Bu da tamamilə git kimi eyni işə yarayır.

Github, Bitbucket serverlərdir, hansı ki, yaratdığımız proyektləri git nəzarət sistemi vasitısilə buraya daxil edirik. Və bu serverdə digər team əməkdaşları mənim yaratdığım proyektin Github,Bitbucket-dəki repository-ni görə bilər. Fork edərək üzərində dəyişiklik edə və sonra mənə pull request edə bilərlər.

1. Git ilə Github arasındakı fərq nədir?

Cavab: Git proyekt üzərində dəyişikliklərə nəzarət edilməsi, bir çox insanların qrup şəklində fayl üzərində işləməsi üçün nəzarət sistemidir. Github serverlərdir, hansı ki, yaratdığımız proyektləri git nəzarət sistemi vasitısilə buraya daxil edirik. Və proyektlərimiz bizim github-da yaratdığımız repository-lərə əlavə olunurlar. Git-in bir çox operatorları var(git add, commit, push,clone və s.)

1. Sıra ilə izah edin hansı nə iş görür? Init, clone, pull, push, checkout, fork

Cavab: Init- GitHub-da yeni repository yaratmaq üçündür.

Clone- ilk dəfə proyekti url-dən çəkəndə istifadə olunur.

Pull-ilk dəfə istisna olmaqla bir neçə dəfə repository-dən url-dən çəkmək lazım olduqda istifadə olunur.

Push- url-dən çəkilən proyekt üzərində edilən dəyişiklikləri git add,commit etdikdən sonra Github-a göndərmək üçün istifadə olunur.

Checkout- git branch tərəfindən yaradılan branch-lar arasında dəyişiklik etmək istədikdə istifadə olunur.

Fork- hər hansı şəxs tərəfindən yaradılmış proyekti(repository-ni) özümüzə çəkmək üçün istifadə olunur. Biz onun contributor-u oluruq .

1. TCP Protokolu dedikdə nə başa düşürsünüz?

Cavab: TCP protokolu adından göründüyü kimi qaydalar toplusudur. TCP protokolu vasitəsilə mesajları, məlumatları, faylları və s -ni göndərmək üçün istifadə olunur. Http, SMPP protokolları da arxada TCP protokolu istifadə edirlər məlumat göndərmək üçün. TCP protokolu informasiyanı paketlərlə göndərir, qəbul edir.

1. Java dili olmasaydı TCP anlayışı mövcud olardımı? Yəni TCP ilə işləyə bilməyimiz üçün Java dili mütləqdirmi?

Cavab: Təbii ki, TCP protokolu Java dili olmasaydı da mövcud olardı. Çünki, Bu bir protokoldur. Proqramlaşdırma dilləri bu protokoldan istifadə edib, məlumat göndərirlər. Javada da TCP-dən istifadə etmək üçün class-lar mövcuddur. TCP ilə işləyə bilməyimiz üçün Java dili mütləq deyil.

1. TCP Client dedikdə nə başa düşürsünüz?

Cavab: TCP Client dedikdə TCP protokolundan istifadə edərək məlumat göndərəcək şəxs(müştəri)-dir. TCP client Java dilində Socket class-nın obyektini yaradan və onun konstruktoruna ip və port göndərən və göndərəcəyi məlumatı oxuyan, serverə yazan kodlar toplusu kimi başa düşürəm.

1. TCP Server dedikdə nə başa düşürsünüz?

Cavab: TCP Server dedikdə TCP protokolundan istifadə edərək məlumatı TCP client tərəfindən qəbul edəcək serverdir.TCP Server Java dilində ServerSocket class-nın obyektini yaradan və onun konstruktoruna port göndərən, bu obyektin dəyişəninin köməyilə socketi gözləyən və göndərilən məlumatı oxuyan, lazımi directory-ə yazan kodlar toplusu kimi başa düşürəm.

1. TCP üzərində qurulmuş başqa hansı protokolları bilirsiniz?

Cavab: Http, SMPP protokolları da arxada TCP protokolu istifadə edirlər məlumat göndərmək üçün. TCP protokolu informasiyanı paketlərlə göndərir, qəbul edir.

1. Thread nədir?

Cavab: Thread Java dilində müəyyən əməliyyatları(task)-ları bir-bir,sətirbəsətir yox bir yerdə yerinə yetirmək istəyiriksə, bu zaman Thread-lərdən istifadə olunur. Yaradılmış bir neçə Thread vasitəsilə bir əməliyyatı Thread-lərin eyni zamanda işə başlaması ilə daha tez yerinə yetirə bilərik. Thread anlayışı Əməliyyat sistemi, texniki səviyyədə Core-larla əlaqəlidir.

1. Runnable nədir?

Cavab: Runnable bir Interface-dir. Runnable interface-ni implement edib yaratdığımız class Thread class-ın targetinə mənimsədilir. Runnable interface-ni implement edən yaratdığımız class-da thread-in yaradılması üçün Override olunan run metodu olur. Və Thread t=new Thread(new MyServer());

t.start; vasitəsilə thread yaranmış olur.

1. Threadləri yaradarkən extends Thread daha üstündür yoxsa implements Runnable

Cavab: Thread-ləri yaradərkən implements Runnable etmək daha üstündür. Çünki, extends Thread edəndə class-ın extends elemek xassəsini itirmiş oluruq. Bu da bizə işimizə mane ola bilər. Bu nöqteyi-nəzərdən implement edib direct Thread-in targetine menimsetmek daha meqsedeuygundur.Və extends eləmək xassəsini də itirməmiş olacağıq.

1. race condition nədir və race conditionın qarşısını almaq üçün nə etməliyik?

Cavab: Race condition yarış vəziyyəti deməkdir. Metodlar synchronized olmadıqda run metodunu start vasitəsilə bir çox thread çağırdıqda, bunlar arasında race condition baş verir. Hansı thread qabağa keçdi keçdi. Bəzi thread-lər eyni milli saniyədə işləyəcək. Məsələn: a-nı 1 vahid 100 dəfə artiracaqdiysa, aralarında race condition olduğuna görə eyni zamanda işləyənlər hesabına artıq 100 ola da, olmaya da bilər

1. Synchronization dedikdə nə başa düşürsünüz?

Cavab: Synchronization lock üçün istifadə olunur. synchronized(obyekt dəyişəni){ } istifadə olunaraq lock qoyulur. Digər bir metodda olan synchronized(obyekt dəyişəni){ } bu keyword-da eyni monitor istifadə olunarsa, digər thread-lərin o metoddakı synchronized(obyekt dəyişəni){ } bu yazılan hissəyə girişi block olunur(lock). 1-ci metoddakı synchronized(obyekt dəyişəni){ } -dakı thread işini bitirən kimi bura daxil olan thread ilə digər metoddakı synchronized(obyekt dəyişəni){ }-a daxil olan thread arasında race condition baş verir. Hansı daxil olarsa digərinin synchronized hissəsin lock edir. Monitorları eyni olduğuna göre bunlar baş verir. Monitor yəni ki, synchronized(obyekt dəyişəni) buradakı mötərizə daxilindəki obyekt dəyişəni başa düşülür. Bu mötərizə daxilində this yazıldıqda başa düşülməlidir ki, metodun olduğu class-ın obyektidir. Lock qoymaq üçün digər obyektlər də yaradıla bilər.

1. Public syncrhronized void foo(){} buradakı synchronized keywordü nə üçün istifadə olunur?

Cavab:

1. monitor nədir, lock nədir, synchronized(obyekt) nə üçün istifadə olunur

Cavab: Lock thread-lərin metodlara girişinə block(məhdudiyyət) qoyulur. synchronized(obyekt dəyişəni){ } istifadə olunaraq lock qoyulur. Digər bir metodda olan synchronized(obyekt dəyişəni){ } bu keyword-da eyni monitor istifadə olunarsa, digər thread-lərin o metoddakı synchronized(obyekt dəyişəni){ } bu yazılan hissəyə girişi block olunur(lock). 1-ci metoddakı synchronized(obyekt dəyişəni){ } -dakı thread işini bitirən kimi bura daxil olan thread ilə digər metoddakı synchronized(obyekt dəyişəni){ }-a daxil olan thread arasında race condition baş verir. Hansı daxil olarsa digərinin synchronized hissəsin lock edir. Monitorları eyni olduğuna göre bunlar baş verir. Monitor yəni ki, synchronized(obyekt dəyişəni) buradakı mötərizə daxilindəki obyekt dəyişəni başa düşülür. Bu mötərizə daxilində this yazıldıqda başa düşülməlidir ki, metodun olduğu class-ın obyektidir. Lock qoymaq üçün digər obyektlər də yaradıla bilər.

1. deadlock nədir və bunun qarşısını almaq üçün nə etməliyik?

Cavab: Deadlock Məsələn: run 2 metodumuz, 2 monitorumuz var. Hər iki metodda lock istifadə edirik. Aşağıdakı kimi:

private static class ThreadDemo1 extends Thread {

public void run() {

synchronized (Lock1) {

System.out.println("lock 1");

synchronized (Lock2) {

System.out.println("lock 2");

}

}

}

}

private static class ThreadDemo2 extends Thread {

public void run() {

synchronized (Lock2) {

System.out.println("lock 2");

synchronized (Lock1) {

System.out.println("lock 1");

}

}

}

}

Deməli, 1-ci thread ThreadDemo1-in run metoduna daxil olursa, synchronized (Lock1) -bura daxil oldu. İşini qurtarana kimi ThreadDemo2-dəki run metoduna girən thread synchronized (Lock2) -dən rahatlıqla keçir, lakin synchronized (Lock1)-a daxil ola bilmir. Digəri də bu cür baş verir. Deadlock-dan çıxış yoxdur. Deadlock-un qarşısını almaq üçün local lock-lardan istifadə, ümumiyyətlə lock-dan istifadə etməmək, bir neçə lock-dan istifadə etməmək, lock daxilində digər metodları çağırmayın(həmin metodlar deadlock-a sala bilər).

1. volatile nə üçün istifadə olunur?

Cavab: Volatile- thread-lar bir neçə core-dan istifadə etdikdə bir core-dakı thread-lərin digər core-dakı thread-lərin etdiyi əməliyyatlardan xəbəri olmur. Bir çox core-lardan istifadə edən thread-lərin bir-birini görməsi üçün volatile istifadə olunur.

1. Threadlərdə join nə üçün istifadə olunur

Cavab: ExecutorService-də newSingleThreadExecutor() metodu ilə thread-lər bir-bir yaranırdı. Lakin bu ExecutorService-dan ayrı başqa bir üsul join üsulu vardır.

Thread t=new Thread(new MyThread());

Thread t1=new Thread(new MyThread());

t.start();

t.join();

t1.start();

t1.join();

Yazılmaqla Thread-lər bir-bir yaranır. ExecutorService-dan istifadə etmək daha məqsədəuyğundur.

1. ExecutorService nədir? Hansı növ executor service-ləri tanıyırsınız?

Cavab:ExecutorService- Thread-lərə nəzarət edən bir APİ-dir.

Executors.newSingleThreadExecutor();

Executors.newFixedThreadPool();

Executors.newScheduledThreadPool();

Executors.newCachedThreadPool();

1. invokeAll, invokeAny, shutdown, awaitForTermination hər birini ayrı-ayrılıqda izah edin. Ay müəllim bir dənəsini unutdum eee deməyin araşdırın yazın.Dərsdə izah olunub. Dərsə baxdığınız zaman qeyd etdiyiniz notelarınıza baxın orada tapacaqsınız.

Cavab: **shutdown**- Hər ExecutorService-dən istifadə edib qurtardıqdan sonra shutdown-dan istifadə edirik və proyekt dayanır.

**awaitForTermination**ExecutorService-classının obyektinin metodudur. awaitForTermination əvvəl shutdown deyirik və ExecutorService-imiz dayanır. Yeni ExecutorService yaradıb thread-ləri yaratmadan qabaq awaitForTermination işlədirik. Burada verilən saniyə ərzində gözləyir, sonra ExecutorService yaradır. NÜmunə:

for (int i = 0; i < 10; i++) {

ScheduledExecutorService service = Executors.newScheduledThreadPool(5);

service.schedule(new MyThread(2), 3, TimeUnit.SECONDS);

service.schedule(new MyThread(2), 3, TimeUnit.SECONDS);

service.shutdown();

service.awaitTermination(7, TimeUnit.SECONDS);

System.out.println("Terminated");

}

**invokeAll**-Callable-dan implements edən class-larımızın arrayini yaradıb Arrays.aslist eliyib invokeAll-a göndəririk. Beləliklə, invokeAll Callable-ları bir-bir run edir.

**invokeAny**-Callable-dan implements edən class-larımızın arrayini yaradıb Arrays.aslist eliyib invokeAny-ə göndəririk. Beləliklə, invokeAny Callable-ları ordan burdan run edir. Performans cəhətdən invokeAny daha məqsədə uyğundur.

1. Callable interface-i nə üçün istifadə olunur və Future nədir?

Cavab: **Callable interface**-i Runnable ilə tamamilə eynidir. Override olunan metodun adı fərqlidir Callable-da(call). Callable-ın üstünlüyü Callable-u implements edib yaratdığımız Thread üçün geriyə Future classı return edir . Future classı da thread-lərin haqqında məlumat verən bir class-dır.

**Future**- Future classı da thread-lərin haqqında məlumat verən bir class-dır.

Get

Cancel

isCancelled

isDone

Metodları vardır. Yaranan Threadin Future üçün istifadə etdiyimiz obyekt implements Callable olmalıdır.

Praktiki Tapşırıq:

Proyekt maven-da olacaq və sonda öz git repositorynizə push edəcəksiniz və mənə github repository linkini göndərəcəksiniz proyekti zip edib göndərməyəcəksiniz.

Proyekt açılanda müştəridən ad soyadını istəyəcəksiniz. Müştəri ad soyadını daxil edəcək.

Müştəridən soruşusunuz:

Zəhmət olmasa göndərmək istədiyiniz faylın yerləşdiyi keçidi qeyd edin. Tutaq ki, daxil edirik:

C:/Users/Sarkhan Rasullu/Desktop/qeydlerim.txt

Daha sonra müştəridən yenidən soruşursunuz bu şəkli göndərmək istədiyiniz şəxsin(serverin) ip və portunu daxil edin:

Biz bu qaydada daxil edəcəyik:

localhost:5678

Iki nöqtə olsun mütləq. Siz bu mesajı split edəcəksiniz və sizə lazım olan ip və portu əldə edəcksiniz:

String[] arr = str.split(“:”);

String ip = arr[0];

int port = Integer.parseInt(arr[1]);

Və bu ip port-a daxil olub həmin faylı göndərəcəksiniz göndərdikdən sonra da mesaj çapa verəcəksiniz ki, uğurla göndərildi.