TALABALARNI ISTISNOLARNI QAYTA ISHLASHNI O‘RGATISHDA JAVA DASTURLASH TILIDAN FOYDALANISH

SH.SH.Mirzanova

Samarqand temir yo‘l texnikumi maxsus fan o‘qituvchisi

ANNOTATSIYA

Java dasturlash tilida istisno tushunchasi, ular bilan ishlash, istisno turlari, Java dasturlash tilida istisnolarni “tutish”, istisnolarni qayta ishlash.

Kalit so‘zlar: *Java, istisno,ob’yekt, try, catch, throw, throws va finally.*

Yaratilgan dasturlarning ishlash jarayonlarida ba’zan kutilmagan xatolar vujudga kelishi mumkin. Bunday hollarda dasturda favqulotda to‘xtash jarayoni yuz beradi, bu esa dasturdan foydalanishni ancha qiyinlashtiradi. Kutilmagan xatolar bilan ishlash uchun ko‘pchilik dasturlash tillarida istisnolarni qayta ishlash mexanizmi kiritilgan. Dasturlashtirishni o‘rganayotgan talabalarga bu mexanizm ancha tushunarsiz bo‘ladi. Biz bu yerda Java dasturlash tili orqali istisnolarni qayta ishlash mexanizmini qarab chiqamiz. *Istisno* – dasturning bajarilishida kodlar ketma-ketligida kelib chiqadigan halokatli holat (buzilish holati) dir. Boshqacha aytganda, istisno – bajarilish vaqtidagi xatolik. Istisnolarni qayta ishlashni qo‘llab-quvvatlamaydigan mashina tillarida odatda xatoliklar kodi yordamida sun’iy tarzda tekshirilishi va qayta ishlanishi kerak bo‘ladi. Bunday yondashuv yetarlicha murakkab va ko‘p ish hajmini oladi. Java dasturlash tilida bu muammolar qarab chiqilgan, bajarilish vaqtida xatoliklar kelib chiqqanda boshqaruv istisnolarni qayta ishlash oqimiga uzatiladi.

Java dasturlash tilida istisno – bu kodning biror qismida kelib chiqqan istisno (ya’ni xatolik) holatini tavsiflovchi obyekt. Istisno holati kelib chiqqanda, shu istisnoni tasvirlovchi obyekt yaratiladi va xatolikni chaqirgan usulga uzatiladi. O‘z

navbatida usul shu istisnoni o‘zi qayta ishlashi yoki boshqa usulga uzatishi mumkin. Har qanday holda biror nuqtada istisno “tutiladi” va qayta ishlanadi.

Java dasturlash tilida istisnolar beshta – try, catch, throw, throws va finally kalit so‘zlari bilan boshqariladi. Istisno holatlarga nisbatan nazorat qilinishi kerak bo‘lgan operatorlar try bloki ichiga yoziladi. try bloki ichida istisno holati yuz bersa, u holda bu istisno holat shu blok tomonidan yuborilgan deyiladi. Kod bu istisnoni tutib oladi (catch operatori yordamida) va biror ratsional usulda qayta ishlaydi. Javaning bajaruvchi tizimi yordamida generatsiya qilingan istisnolar avtomatik tarzda yuboriladi. Istisnolarni “qo‘lda” tutish uchun throw kalit so‘zidan foydalaniladi. Usuldan yuborilgan har qanday istisno shu usulning aniqlanishidagi sarlavha satridagi throws kalit so‘zi yordamida aniqlanishi kerak. try-blokidan qaytishdan oldin bajarilishi kerak bo‘lgan har qanday kod try{...}-catch{...}-finally{...} blokli konstruksiyada ko‘rsatilishi kerak bo‘lgan finally-blokiga yoziladi.

```
try{
// xatolik ustida nazorat uchun kod bloki
}
catch(ExceptlonType1 exOb) {
// ExceptlonType1 uchun istisnolarni qayta ishlash
}
catch(ExceptlonType2 exOb) {
// ExceptlonType2 uchun istisnolarni qayta ishlash
}
//. . .
finally{
// try blokidan qaytgandan keyin qayta ishlash bloki
}
```

Bu yerda *ExceptionType – yuz bergan istisno holati turi; exOb – shu istisnoning obyekti,* finally-blok – shart bo‘lmagan parametr.

Masalan, quyidagi kichik dasturda nolga bo‘linish xatoligini chaqiruvchi ifodani

o‘z ichiga oladi.

```
class Yexs0{
public static void main(String args[]){
int d=0;
int a=42/d;
} }
```

Javaning bajaruvchi tizimi nolga bo‘linish belgisini topsa, u yangi turdagi obyekt yaratadi va uni qayta ishlashga uzatadi. Bu Yexs0 ning bajarilishini to‘xtashiga majbur qiladi, u istisnoni qayta ishlovchi tomonidan tutib qolinishi kerak. Ko‘rsatilgan misolda istisno Java ning bajaruvchi tizimi qayta ishlovchisi tomonidan tutib qolinadi va qayta ishlanadi. Dastur tomonidan ushlab qolinmagan har qanday istisno nihoyatda jimlik holatidagi qayta ishlovchi tomonidan ushlab qolinadi. Bu qayta ishlovchi istisnoni tavsiflovchi satrni (ekranga) chiqaradi, istisno yuz bergan nuqta chop etiladi va dasturni tugatadi.

Quyida standart JDK ning Java-interpretatorida bajarilganda quyidagi natijani generatsiya qiladi:

```
Java.lang.ArithmeticException:/byzero//xatolik haqidagi xabar
atexs0,main(Yexs0.Java:4)
```

Natijada quyidagi elementlarning qanday qo‘shilganligiga e’tibor bering: sinf nomi (Exc0), usul nomi (main()), fayl nomi (Yexs0.java) va satr nomeri. Bunday tashqari yuz bergan xatolikni yanada ochiqroq tavsiflaydigan istisno turi ArithmeticException kabi nomlangan Exception ning qismiy sinfi ekanligini ko‘rish mumkin.

```
class Exc1{
static void subroutine(){
int d=0; int a=10/d; }
public static void main(String args []){
Excl.subroutine();
} }
```

Natijada (jimlik holatidagi istisnolarni qayta ishlovchi tomonidan olingan)

quyidagilar aks etadi:

```
Java.lang.ArithmeticException:/byzero//(soobsheniyeoboshibke)
atExcl.subroutine(Excl.Java:4)//(trassapolnogosteka
atExcl.main(Excl.Java:7)//vizovov)
```

Java ning bajaruvchi tizimining jimlik holatidagi istisnolarni qayta ishlovchisi otladka uchun foydali bo‘lsa-da, dasturchi ba’zan istisnolarni o‘zi mustaqil qayta ishlashni hohlaydi. Bu esa ikki afzallikni ta’minlaydi: birinchidan, xatolikni fiksirlashga imkon beradi, ikkinchidan, dasturning avtomatik tugallanishidan saqlaydi. Agar dasturingiz har safar xatolik kelib chiqqanida tugallansa va bir qancha istisno tavsiflarini chop etsa, u holda dasturingizdan foydalanuvchilar norozi bo‘ladilar.

Bu holatni bartaraf etish juda ham sodda!

Bajarilish vaqtidagi xatolarni kuzatish va qayta ishlash uchun shunchaki kodga try blokini qo‘shish yetarli. try blokidan keyin tutilishi va qayta ishlanishi kerak bo‘lgan istisno turini aniqlaydigan catch-bloki qo‘shiladi.

Quyidagi dasturda try va catch bloklarinidan foydalanish naqadar sodda ekanligini ko‘rish mumkin, dasturda “nolga bo‘linish” xatoligini generatsiya qiladigan ArithmeticException istisno turi qayta ishlanadi.

```
class Yexs2{
public static void main{String args []}{
intd b,a;
try{                                    //kod blokini nazorat qilish
d-0;
a=42/d;
System.out.println("Bu chop etilmaydi");
}catch(ArithmeticExceptione){
System.out.println("Nolga bo‘linish.");
}
System.out.println("catch operatoridan so‘ng,");
} }
```

Bu dastur quyidagi natija beradi:

Nolga bo‘linish.

Catch operatoridan so‘ng.

try bloki ichidagi println()operatoriga murojaat hech qachon murojaat qilinmasligiga e’tibor bering. Istisnoli holat kelib chiqqach, dastur boshqaruvi try blokidan catch blokiga uzatiladi. Shunday qilib, “Bu chop etilmaydi” satri ekranga hech qachon chiqarilmaydi. catch operatori bajarilishi tugallanishi zahoti boshqaruv try/catch ning to‘liq mexanizmidan keyingi satrga uzatiladi.

FOYDALANILGAN ADABIYOTLAR RO‘YXATI

1.*Bryus Ekkel. Filosofiya Java. 3-e izd. SPb.: Piter, 2.Jerzdev S.V. Java 2 Micro Edition. ITLab, NNGU, VMK.*