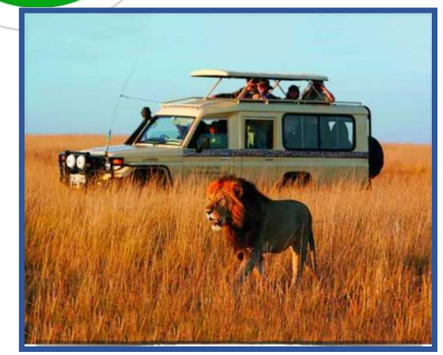
Exception



Her sey olmasi gerektigi gibi..
No exception

Sorun var ama halledilebilir..
No Exception

Sorun var ve halledilemez
Throw Exception

Java'da bir program calistirildiginda, farkli sorunlar olusabilir.

- Programcilarin yazdigi kodlarda hata olabilir
- Kullanicidan istenen degerlerde uygun olmayan deger girilebilir
- Internet baglantisinin kesilmesi gibi ongorulemeyen hatalar olabilir

NOT : Bir program calistirildiginda, Java cozemedigi bir sorunla karsilastiginda calismayi durdurur (stops execution) ve "throws an exception"

Exception

Java karsilastigi sorunu ve sorunla karsilastigi kod satirini bize rapor eder

```
 5 public class Go {
 6
 7     public static void main(String[] args) {
 8     System.out.println("");
 9             int a=10;
10             int b= 0;
11
12             System.out.println(a/b);
13
14 }
15 }
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at _00_example.Go.main(Go.java:12)
```

Exception turu

Sorunun oldugu satir

Exception'in kaynagi

Exception

Sorunu cozmek icin try - catch block kullaniriz.

Try blogu tek basina calismaz.

Try blogundan sonra mutlaka catch block(lari) veya finally block olmalidir.

```
public class Go {

    public static void main(String[] args) {
            int a=10;
            int b= 0;

            try {
                System.out.println(a/b);
            } catch (ArithmeticException e) {
                System.out.println("Bolme isleminde payda sifir olamaz");
            }
    }
}
```

Sorun cikmazsa yapacagi islem
(Bizim asil yapmak istedigimiz islem)

beklenenExceptionTuru

Beklenen exception turu
gerceklendiginde calisacak kodlar

Exception

Catch block'da kullanilan "e" nin gorevi

- Catch block'da yazdigimiz Exception ismi class adi (Data turu) , "e" ise variable ismidir.
- e. yazinca ilgili exception class'indan kullanabilecegimiz method'lari gorebiliriz.

```
catch (FileNotFoundException e) {

      System.out.println("Dosya okunamiyor" + e.getMessage());
  }
```

Exception'in kaynagini gosterir

```
catch (FileNotFoundException e) {

      e.printStackTrace();
      System.out.println("Dosya okunamiyor" );
  }
```

Detayli Exception'in raporu gosterir

Exception

File Input/Output Exceptions

```
 5 public class Go {
 6
 7     public static void main(String[] args) {
 8         FileInputStream fis=new FileInputStream("\\src\\_00_example\\file");
 9     }
10 }
```

Java'dan bir dosya okumasini veya bir dosyaya yazmasini istedigimizde, Java olasi problemleri ongorur ve bizden cozum ister.

Buradaki CTE, kodumuzda bir hata oldugu icin degil yazdigimiz kod calistiginda olusabilecek olasi okuma hatalarinda ne yapilacagina karar vermek icindir.

Exception

Muhtemel sorunlar birden fazla ise;

1) Icice try-catch bloklari kullanilabilir.

```
public class Go {

    public static void main(String[] args) {
        FileInputStream fis = null;
        //You may use nested try-catch block
        try {
            fis = new FileInputStream("C:\\Users\\lenovo\\eclipse-workspac

            int k = 0;

            try {
                while((k = fis.read()) != -1) {

                    System.out.print((char)k);
                }
            } catch (IOException e) {
                System.out.println("Dosya okunamiyor");
            }
        } catch (FileNotFoundException e) {
            System.out.println("Dosya silinmis veya dosya yolu hatali");
        }
    }
}
```

Exception

2) Tek try- multiple catch kullanilabilir.

```
public class Go {

    public static void main(String[] args) {
        FileInputStream fis = null;

        try {
            fis = new FileInputStream("C:\\Users\\lenovo\\eclipse-workspac

            int k = 0;


            while((k = fis.read()) != -1) {

            System.out.print((char)k);

        }
        } catch (FileNotFoundException e) {
                System.out.println("Dosya okunamiyor");
            }
           catch (IOException e) {
            System.out.println("Dosya silinmis veya dosya yolu hatali");
        }
    }
}
```

Birden fazla catch block kullanilacaksa yazilacak exception'larin sirasi onemlidir.

Birbiri ile parent-child iliskisi olan exception'lar ise once child olan yazilmalidir. Aksi durumda child exception kullanilmaz olur.

Exception

3) Eger tum exception'lari iceren bir exception varsa sadece onu yazabiliriz.

```
public class Go {

    public static void main(String[] args) {
        FileInputStream fis = null;

        try {
            fis = new FileInputStream("C:\\Users\\lenovo\\eclipse-wo

            int k = 0;


            while((k = fis.read()) != -1) {

            System.out.print((char)k);

        }
        }
           catch (IOException e) {
            System.out.println("Dosya okuma problemi var. "
                    + "Dosya silinmis veya path hatali olabilir");
        }
    }
}
```

Exception Types

1) Compile Time (Checked) Exceptions

   Kod yazildiginda Java'nin ongordugu olasi sorunlardir. Java olasi bir problem gordugunde kirmizi cizgi ile bizi uyarir. (Not: Her kirmizi cizgi exception degildir.)

   (FileNotFoundException,IOException)

2) RunTime (Unchecked) Exceptions

   Kod calistirildiginda ortaya cikan exception'lardir.

   (ArithmeticException)

Exception

2) NullPointerException

- null objesini uygun olmayan bir komutta kullanirsaniz Java NullPointerException verir.
- NullPointerException run time exception'dir.

```
public class Gogo {

    public static void main(String[] args) {
        String str="";
        System.out.println(str.length()); // 0

        str=null;
        System.out.println(str.length()); // RTE
    }

}
```

```
Exception in thread "main" java.lang.NullPointerException
        at _00_example.Gogo.main(Gogo.java:10)
```

Exception

3) ArrayIndexOutOfBoundsException

- Array veya List'de olmayan bir index icin islem yapmak isterseniz Java ArrayIndexOutOfBoundsException verir.

- ArrayIndexOutOfBoundsException run time exception'dir.

```
public static void main(String[] args) {
    int arr[]= {1,2,3};
    System.out.println(arr[0]); // 1


    System.out.println(arr[4]); // Exception
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
        at _00_example.Gogo.main(Gogo.java:10)
```

Exception

4) ClassCastException

- Bir datayi casting yapilamayacak bir dataya cevirmek istediginizde ClassCastException verir.

- ClassCastException run time exception'dir.

```
public static void main(String[] args) {
    Integer sayi= 10;
    System.out.println(sayi); // 10

    String str= sayi; //CTE

    String str2= (String)sayi; // CTE

    Object sayi2=40;
    String str3=(String)sayi2; // Exception

}
```

```
Exception in thread "main" java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.Str
        at _00_example.Gogo.main(Gogo.java:14)
```

Exception

5) NumberFormatException

- Sayi olmayan bir String'i sayiyaya cevirmeye calisirsaniz Java NumberFormatException verir.
- NumberFormatException run time exception'dir.

```
public static void main(String[] args) {
    String str= "123456";

    int sayi =Integer.parseInt(str);
    System.out.println(sayi+10); // 123466

    str="123a4";
    sayi =Integer.parseInt(str);
    System.out.println(sayi+10); // Exception
}
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "123a4"
        at java.lang.NumberFormatException.forInputString(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at _00_example.Gogo.main(Gogo.java:12)
```

Exception

Soru 1)

Kullanicidan carpma yapmak icin bir String isteyin. Kullanicinin girdigi String sadece sayilardan olusuyorsa sayiyi 2 ile carpip sonucu yazdirin.
Kullanici sayilardan olusmayan bir String girerse “Girdiginiz String sayiya cevrilemez” yazdirin.

Soru 2)

String str[ ],Urun isimlerini tuttugumuz bir Array olsun. Kullanicidan istedigi urunun sirasini isteyin ve istedigi urunu yazdirin.
Kullanici Array’de olan urun sayisindan buyuk bir sira no girerse “Girdiginiz sira urun sayisindan buyuk” yazdirin.

Exception

6) illegalArgumentException

Soru: Kullanicidan yasini girmesini isteyin. Kodunuzu kullanici sifirdan kucuk bir sayi girerse Exception verecek sekilde yazin.

```
public class Example {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        System.out.println("Konsolda gormek icin yasinizi girin");
        int yas = scan.nextInt();

        try {
            if(yas<0) {
                throw new IllegalArgumentException();
            }
        }catch(IllegalArgumentException e) {
            System.out.println(e);
            System.out.println("Yas icin negatif deger giremezsiniz...");
    }
        System.out.println(yas);
        scan.close();
    }
}
```

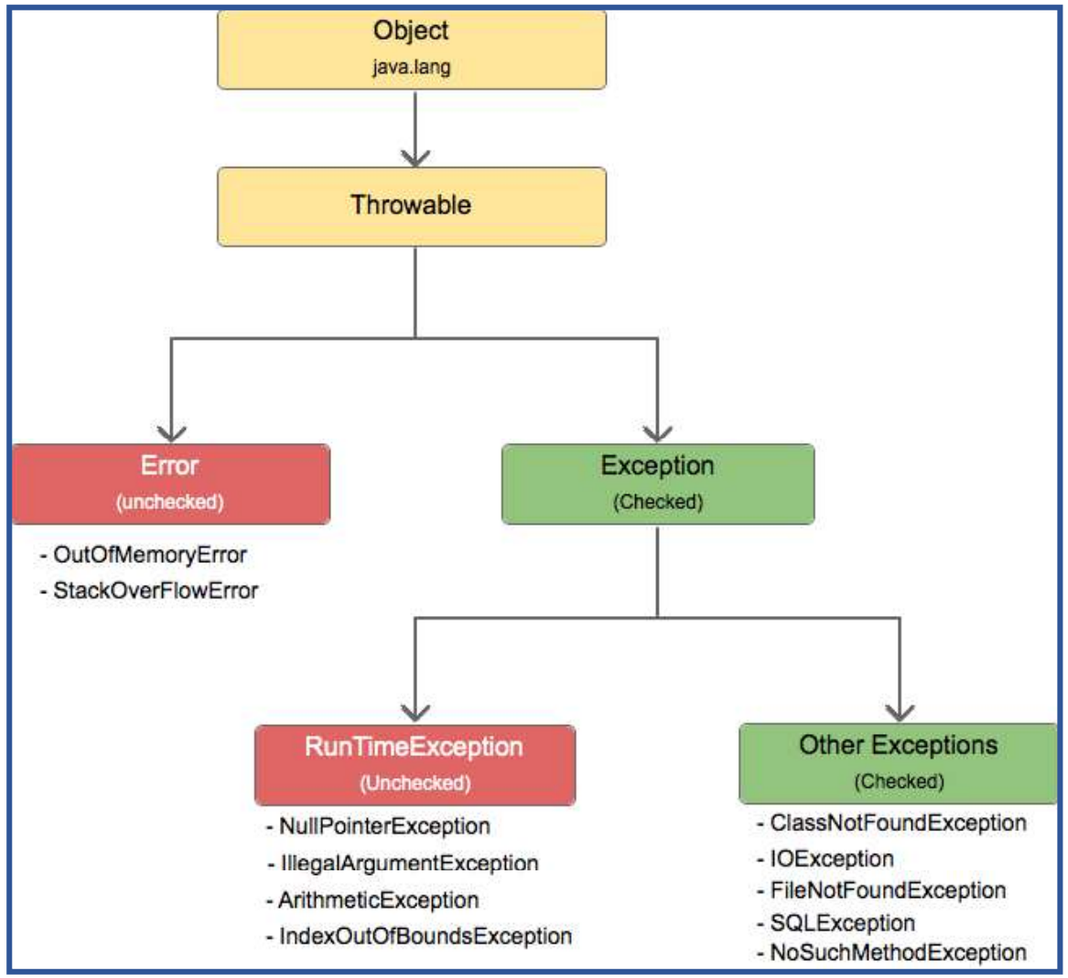
```
-20
java.lang.IllegalArgumentException
Yas icin negatif deger giremezsiniz...
-20
```

Exception

throws

- throws keyword “checked exceptions” icin kullanilir.
- throws keyword, exception handle yapilmak istenmiyorsa kullanilir. (Exception olusunca program calismasi durur)
- throws keyword’den sonra, aralarina virgul konularak, birden fazla exception yazilabilir
- throws keyword method body icinde kullanilamaz, kullanilacaksa method isminin oldugu satirda yazilmalidir.
- throws keyword’den sonra birden fazla exception kullanilacaksa ve yazilan exception’lar arasinda parent child iliskisi varsa , child exception yazilabilir ama tavsiye edilmez. Cunku parent exception tum durumlari kapsayacaktir. (Hedef farkli durumlar icin aciklama yazip handle etmek olmadigindan, bir exception’in calismasi yeterlidir)

Exception


Object
java.lang
Throwable
Error
(unchecked)
- OutOfMemoryError
- StackOverFlowError
Exception
(Checked)
RunTimeException
(Unchecked)
- NullPointerException
- IllegalArgumentException
- ArithmeticException
- IndexOutOfBoundsException
Other Exceptions
(Checked)
- ClassNotFoundException
- IOException
- FileNotFoundException
- SQLException
- NoSuchMethodException

Exception

THROW	THROWS
1- throw keyword kontrollu olarak bir exeption throw etmek icin kullanilir	1- throws keyword bir veya daha fazla exeption'i deklare etmek icin kullanilir
2- throw keyword ile sadece bir exeption throw edilebilir	2- throws keyword bir veya daha fazla exeption throw edilebilir
3- throw keyword method icinde kullanilir	3- throws keyword method signature ile kullanilir
4- throw keyword yazdiktan sonra variable yazilir (Orn: new IllegalArgumentException();)	4- throws keyword yazdiktan sonra exception class ismi yazilir (Orn: FileNotFoundException)

Exception Finally Block

- Finally block try-catch blogu ile kullanilir.
- Finally block, her durumda calisir
- Finally block cloud veya database ile connection'i bitirme veya uzerinde calisilan dosyayi kapatma gibi islemler icin kullanilir.

```
public static void main(String[] args) {

    int sayi1=10;
    int sayi2=0;
    try {
    System.out.println(sayi1/sayi2);
    } catch (ArithmeticException e) {
        System.out.println("Kodda hata var... ");
        e.printStackTrace();
    } finally {
        System.out.println("connection'i durdur...");
    }
    }
```

- Finally block catch blogu olmadan sadece try ile de kullanilabilir.
- Bu durumda catch blogu olmadigindan Java throws exception ardindan finally ile istedigimiz islemi yapar

Exception Finally Block

Asagidaki kod calistirildiginda konsolda ne yazdirir?

```
String s = "";
try {
    s += "t";
} catch (Exception e) {
    s += "c";
} finally{
    s += "f";
}
s += "a";
System.out.print(s);
```

Exception Finally Block

Asagidaki kod calistirildiginda konsolda ne yazdirir?

```
public static void main(String[] args) {

    System.out.println(exceptions());
}

@SuppressWarnings("finally")
public static String exceptions() {
    String result="";
    String v=null;

    try {
        try {
            result=result+"a";
            v.length();
            result=result+"b";
        } catch(NullPointerException e) {
            result=result+"c";
        } finally {
            result=result+"d";
            throw new Exception();
        }
    } catch (Exception e) {
        result=result+"e";
    }
    return result;
}}
```

Exception Tekrar Sorulari

1) try blogu mutlaka catch blogu ile kullanilmalidir. ~~True / False~~

2) finally blogu mutlaka calisir. ~~True / False~~

3) Bir try blogu ile birden fazla catch blogu calistirilabilir. ~~True / False~~

4) Birden fazla catch blok varsa, child olan once yazilmalidir. ~~True / False~~

5) FileNotFoundException nedir?

Programimizda bir dosyayi okumaya calisiyorken, dosya bulunamazsa olusur.
IOException'in subclass'idir.

6) IOException nedir?

Programimizda bir file'a input/output yapiliyorsa ve program calisirken bir problem cikarsa olusur.
Checked exception'dir ve kod yazilirken mutlaka handle edilmelidir.

Exception / Create Custom Checked Exception

```java
public class InvalidEmailIdCheckedException extends Exception {

    private static final long serialVersionUID = 1L;

    public InvalidEmailIdCheckedException(String message) {
        super(message);
    }

}
```

1) Class isminin sonunda "Exception" kullanilir. Bu mecburi degildir ama genel isim verme konsepti boyledir.

2) Bir "checked exception“ olusturacaksak, class’imizi "Exception" class’ina child class yapmaliyiz.

3) "String" parametresi olan bir constructor olusturun ve ilk satirina super(); ekleyin.

Exception / Create Custom Checked Exception

Soru : Yeni bir class olusturalim, icinde mailDogrula(String eMail) olsun. Email adresi @gmail.com veya @hotmail.com icermiyorsa InvalidEmailIdCheckedException versin.

```
public class Test {

    public static void main(String[] args) throws InvalidEmailIdCheckedException {
        mailDogrula("ab@gmail1.com");
    }

    public static void mailDogrula(String eMail) throws InvalidEmailIdCheckedException {
        if (eMail.contains("@hotmail.com") || eMail.contains("@gmail.com") ) {
            System.out.println(eMail);
        } else {
            throw new InvalidEmailIdCheckedException("Email adresi uygun degil");
        }
    }}
```

```
Exception in thread "main" _00_example.InvalidEmailIdCheckedException: Email adresi uygun degil
        at _00_example.Test.mailDogrula(Test.java:16)
        at _00_example.Test.main(Test.java:9)
```

Exception / Create Custom UnCheckedException

```
public class InvalidEmailIdUnCheckedException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public InvalidEmailIdUnCheckedException(String message) {
        super(message);
    }

}
```

1) Class isminin sonunda "Exception" kullanilir. Bu mecburi degildir ama genel isim verme konsepti boyledir.

2) Bir "checked exception“ olusturacaksak, class’imizi "RuntimeException" class’ina child class yapmaliyiz.

3) "String" parametresi olan bir constructor olusturun ve ilk satirina super( ); ekleyin.

Exception / Create Custom UnCheckedException

Soru : Yeni bir class olusturalim, icinde mailDogrula(String eMail) olsun. Email adresi @gmail.com veya @hotmail.com icermiyorsa InvalidEmailIdUnCheckedException versin.

```
public class Test {

    public static void main(String[] args)  {
        mailDogrula("ab@gmail1.com");
    }

    public static void mailDogrula(String eMail)  {
        if (eMail.contains("@hotmail.com") || eMail.contains("@gmail.com") ) {
            System.out.println(eMail);
        } else {
            throw new InvalidEmailIdUnCheckedException("Email adresi uygun degil");
        }
    }}
```

```
Exception in thread "main" _00_example.InvalidEmailIdUnCheckedException: Email adresi uygun degil
        at _00_example.Test.mailDogrula(Test.java:16)
        at _00_example.Test.main(Test.java:9)
```

Exception

Soru 1

Which of the following pairs fill in the blanks to make this code compile? (Choose all that apply)

```
7: public void ohNo() _____ Exception {
8:    _____________ Exception();
9: }
```

A. On line 7, fill in `throw`

B. On line 7, fill in `throws`

C. On line 8, fill in `throw`

D. On line 8, fill in `throw new`

E. On line 8, fill in `throws`

F. On line 8, fill in `throws new`

Cevap : B, D. In a method declaration, the keyword throws is used. To actually throw an exception, the keyword throw is used and a new exception is created.

Exception

Soru 2

Which exception will the following throw?

```
Object obj = new Integer(3);
String str = (String) obj;
System.out.println(str);
```

A. `ArrayIndexOutOfBoundsException`

B. `ClassCastException`

C. `IllegalArgumentException`

D. `NumberFormatException`

E. None of the above.

Cevap : B. The second line tries to cast an Integer to a String. Since String does not extend Integer, this is not allowed and a ClassCastException is thrown.

Exception

Soru 3

What will happen if you add the statement `System.out.println(5 / 0);` to a working `main()` method?

A. It will not compile.

B. It will not run.

C. It will run and throw an `ArithmeticException`.

D. It will run and throw an `IllegalArgumentException`.

E. None of the above.

3) C. The compiler tests the operation for a valid type but not a valid result, so the code will still compile and run. At runtime, evaluation of the parameter takes place before passing it to the `print()` method, so an `ArithmeticException` object is raised.

Exception

Soru 4

Which of the following can be inserted in the blank to make the code compile? (Choose all that apply)

```
public static void main(String[] args) {
  try {
    System.out.println("work real hard");
  } catch (_____________ e) {
  } catch (RuntimeException e) {
  }
}
```

A. Exception

B. IOException

C. IllegalArgumentException

D. RuntimeException

4) C, E. Option C is allowed because it is a more specific type than RuntimeException. Option E is allowed because it isn't in the same inheritance tree as RuntimeException. It's not a good idea to catch either of these. Option B is not allowed because the method called inside the try block doesn't declare an lOException to be thrown. The compiler realizes that lOException would be an unreachable catch block. Option D is not allowed because the same exception can't be specified in two different catch blocks. Finally, option A is not allowed because it's more general than RuntimeException and would make that block unreachable.

Exception

Soru 5

What is the output of the following snippet, assuming a and b are both 0?

```
3:      try {
4:        return a / b;
5:      } catch (RuntimeException e) {
6:        return -1;
7:      } catch (ArithmeticException e) {
8:        return 0;
9:      } finally {
10:       System.out.print("done");
11:     }
```

A. `-1`
B. `0`
C. `done-1`
D. `done0`
E. The code does not compile.
F. An uncaught exception is thrown.

5) E. The order of catch blocks is important because they're checked in the order they appear after the try block. Because ArithmeticException is a child class of RuntimeException, the catch block on line 7 is unreachable. (If an ArithmeticException is thrown in try try block, it will be caught on line 5.) Line 7 generates a compiler error because it is unreachable code.

Exception

Soru 6

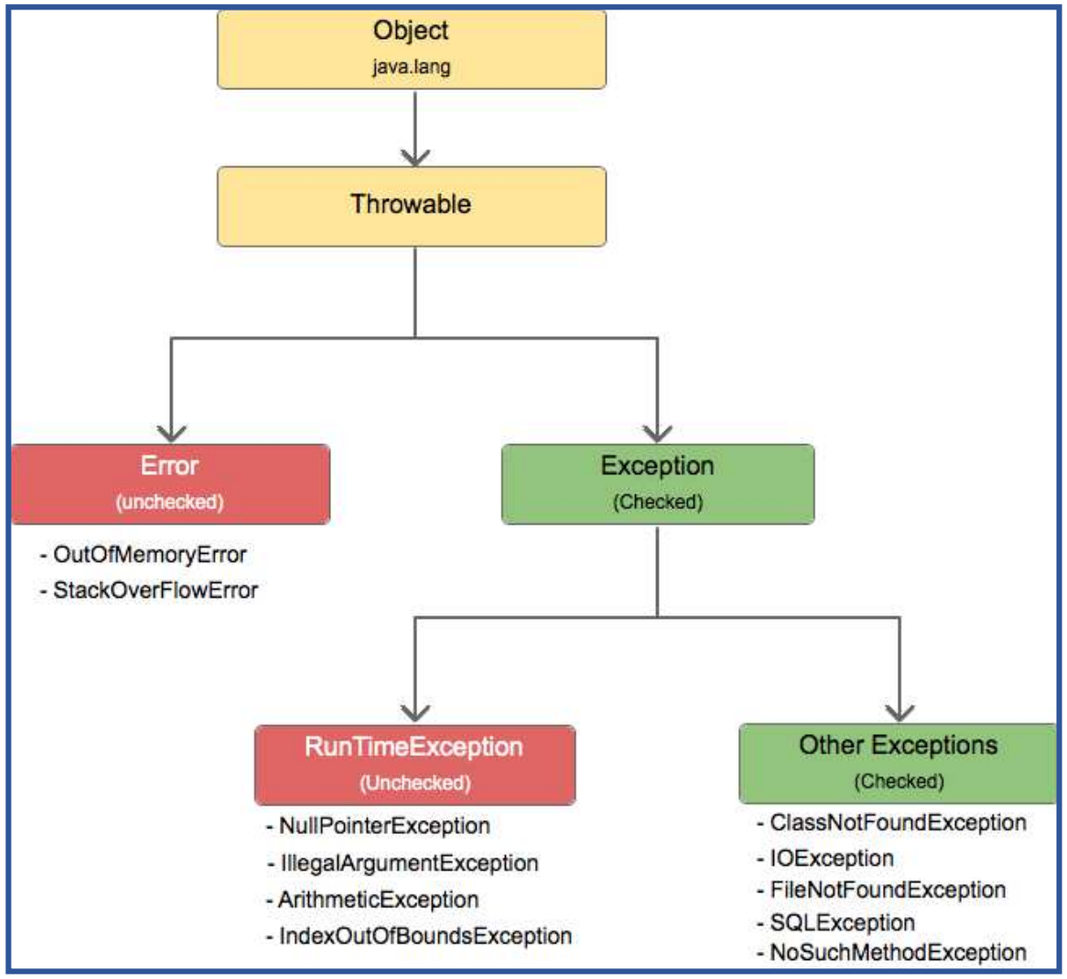
What is the output of the following program?

```
1:  public class Dog {
2:    public String name;
3:    public void parseName() {
4:      System.out.print("1");
5:      try {
6:        System.out.print("2");
7:        int x = Integer.parseInt(name);
8:        System.out.print("3");
9:      } catch (NumberFormatException e) {
10:       System.out.print("4");
11:     }
12:   }
13:   public static void main(String[] args) {
14:     Dog leroy = new Dog();
15:     leroy.name = "Leroy";
16:     leroy.parseName();
17:     System.out.print("5");
18:   } }
```

A. 12
B. 1234
C. 1235
D. 124
E. 1245
F. The code does not compile.
G. An uncaught exception is thrown.

6) E. The parseName method is invoked within main() on a new Dog object. Line 4 prints 1. The try block executes and 2 is printed. Line 7 throws a NumberFormatException, so line 8 doesn't execute. The exception is caught on line 9, and line 10 prints 4. Because the exception is handled, execution resumes normally. parseName runs to completion, and line 17 executes, printing 5. That's the end of the program, so the output is 1245.

Errors


Object
java.lang
Throwable
Error
(unchecked)
- OutOfMemoryError
- StackOverFlowError
Exception
(Checked)
RunTimeException
(Unchecked)
- NullPointerException
- IllegalArgumentException
- ArithmeticException
- IndexOutOfBoundsException
Other Exceptions
(Checked)
- ClassNotFoundException
- IOException
- FileNotFoundException
- SQLException
- NoSuchMethodException

Errors

- Error “throwable” class’inin bir alt class’idir. Errors, sistem kaynaklarının eksikliği nedeniyle ortaya çıkan kritik koşullardır ve yazacagimiz kodlarla handle edilemez.
- Java error’un oluşumu hakkında herhangi bir bilgiye sahip olmadığı için hatalar her zaman unchecked’dirler.
- Errors, her zaman runtime’da ortaya çıkar.
- Error’un ortaya çıkmasının sonucu, programın olağandışı bir şekilde sonlandırılmasıdır.

Errors

Karşılaştırma Tablosu

Karşılaştırma için temel	Hata	İstisna
Temel	Hata, sistem kaynaklarının eksikliğinden kaynaklanır.	Kod nedeniyle bir istisna ortaya çıkıyor.
Kurtarma	Bir hata düzeltilemez.	Bir istisna kurtarılabilir.
Anahtar kelimeler	Bir hatayı program koduyla çözmenin bir yolu yoktur.	İstisnalar, "try", "catch" ve "throw" üç anahtar sözcüğü kullanılarak ele alınır.
sonuçlar	Hata tespit edildiğinde program anormal şekilde sonlandırılır.	Bir istisna tespit edildiğinde, karşılık gelen "atma" ve "yakalama" anahtar sözcükleri tarafından atılır ve yakalanır.
Türleri	Hatalar denetlenmeyen tür olarak sınıflandırıldı.	İstisnalar kontrol edilmiş veya kontrol edilmemiş tip olarak sınıflandırılır.
paket	Java'da hatalar "java.lang.Error" paketi olarak tanımlanmıştır.	Java'da, bir istisna "java.lang.Exception" içinde tanımlanmıştır.
Örnek	OutOfMemory, StackOverFlow.	İşaretli İstisnalar: NoSuchMethod, ClassNotFound. İşaretlenmemiş İstisnalar: NullPointer, IndexOutOfBounds.

Errors

1) Error'lar handle edilemezler.

2) Programin anormal bir sekilde sonlanmasina sebep olurlar.

3) Error'lar unchecked'dir ve genellikle run time'da olusurlar.

4) Error'lara ornek; Out of Memory Error, Stack over flow error veya System Crash Error

```
public static void main(String[] args) {

    for(int i=0; i<3; i--) {
        System.out.print(i + " ");
    }

}
```

Java Platform Bagimsiz Calisir

"write once run everywhere" sloganini gerceklestiren yapidir.

Jvm, Java programlama dilinde yazdığımız kodların üzerinde çalıştığı programdır.

Hangi platformda yazmis olursak olalim Java Compiler kodlarimizi byte kodlara cevirir, burada JVM devreye girerek kodu daha alt seviyede makine diline dönüştürmektir.

Her işletim sisteminde jvm kendine özgüdür.Her işletim sistemi için o işletim sistemine özgü Jvm'i yüklemelisiniz.