Java Time Machine

Wang Yang

Primary Data Type

Type	Size	Range	Wrapper
boolean	N/A		
char			
byte		[]	
short		[]	
int			
long			
float		$3.4*10^{38}$	
double		$1.7*10^{308}$	
void	N/A	N/A	Void

Conversion

- double d =   10
- From Low Accuracy to High Accuracy: Auto
- int   d = (int) 10.2
- From High Accuracy to Low Accuracy: Cast

Naming of Variables

- Which name is legal ?
- 1var
- $class
- class-var

✓

变量名 = 首字母 + 其余部分

首字母

1、字母
2、下划线 ‘_’
3、 ‘$’ 符号

其余部分

任意多的：
1、数字
2、字母
3、下划线 ‘_’
4、 ‘$’ 符号

Operator

- the operand type of “+” operator could be:

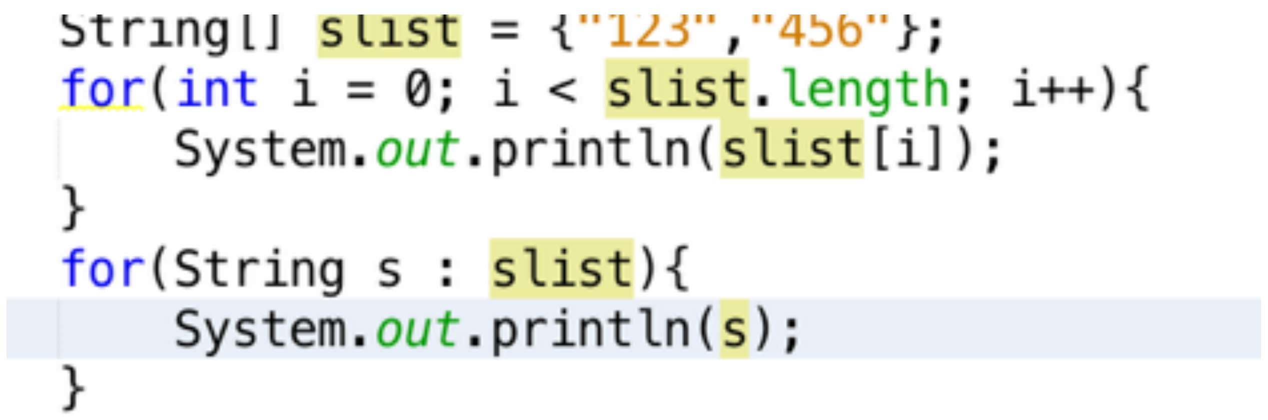
  byte, short, int, long, char, String

Control Flow

- if else 与C/C++不同

  if，while等只接受boolean类型

- String[] sarray, 遍历sarray的for循环



```
String[] slist = {"123","456"};
for(int i = 0; i < slist.length; i++){
    System.out.println(slist[i]);
}
for(String s : slist){
    System.out.println(s);
}
```

main method

- public static void main(String args[]);

OO Techniques

- OO Techniques include:
  - Abstraction
  - Inheritance
  - Polymorphism

Class

- Class includes
  - Field
  - Method
  - Constructor

Field

- How to decorate a Field
  - type
  - static
  - final
  - access control : public/private/protected

Static vs Non-Static

- What's the difference between them
- Give me a example of static value you learned
    - LifeCycle
    - Owner
    - Integer.MAX_VALUE

Method

- How to decorate a method
  - type
  - static
  - final
  - access control : public/private/protected
  - synchronized
  - throws
  - genetic type declare

Method

- Give me a example of static method you learned
    - Arrays.sort
    - Collection.sort

Method

- What’s the method of the signature
    - Method name
    - Number of Parameters
    - Types of Parameters

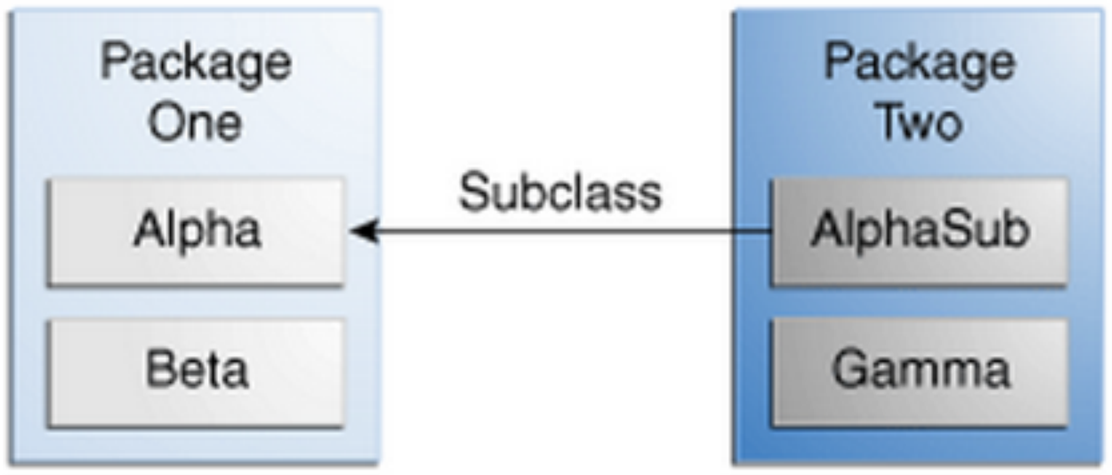
Constructor

- What’s the difference with the normal method
    - No return type
    - same name with the class

Constructor

- default constructor

Access Control



Access to Alpha's member

Modifer	Alpha	Beta	SubAlpha	Gamma
public				
protected				
default				
private				

Package

- list package you've used
    - java.io.*
    - java.util.*
    - java.awt.*
    - javax.swing.*
    - java.net.*

Abstraction

- Type | Implementation
- Two type of Abstraction of Java are:
    - Abstract class
    - Interface

Abstraction

- What’s the difference of Abstract Class and Interface

- Multiple inheritance is allowed for Interface, not for abstract class.
- Abstract class provides part of implementation, while interface has no implementation.

Inheritance

- Between Superclass and Subclass, what the term we call for the subclass redefine the method and the field of the superclass

  - OverWriting

  - Hiding

Inheritance

```
public class House{
    public String className = "House";
    public void showName(){
        System.out.println
            ("The super class: " + className);
    }
}

public class GeorgianHouse(){
    public String className = "GeorgianHouse";
    public void showName(){
        System.out.println
            ("The extended class: " + className);
    }
}
```

Inheritance

✓ A:
GeorgianHouse
House
The extended class: GeoriganHouse
The extended class: GeoriganHouse

B:
GeorgianHouse
GeorgianHouse
The extended class: GeoriganHouse
The extended class: GeoriganHouse

C:
GeorgianHouse
House
The extended class: GeoriganHouse
The super class: House

```
public static void main(String[] args){
    GeorgianHouse gHouse = new GeorgianHouse();
    Hosue house = gHouse;
    System.out.println(gHouse.className);
    System.out.println(house.className);
    gHouse.showName();
    house.showName();
}
```

D:
GeorgianHouse
House
The extended class: GeoriganHouse
The extended class: House

Inheritance

- The Rule of OverWriting:
  - The access rights should be enlarged or unchanged, not be reduced.
  - The return type should be reduced or unchanged, not be enlarged.

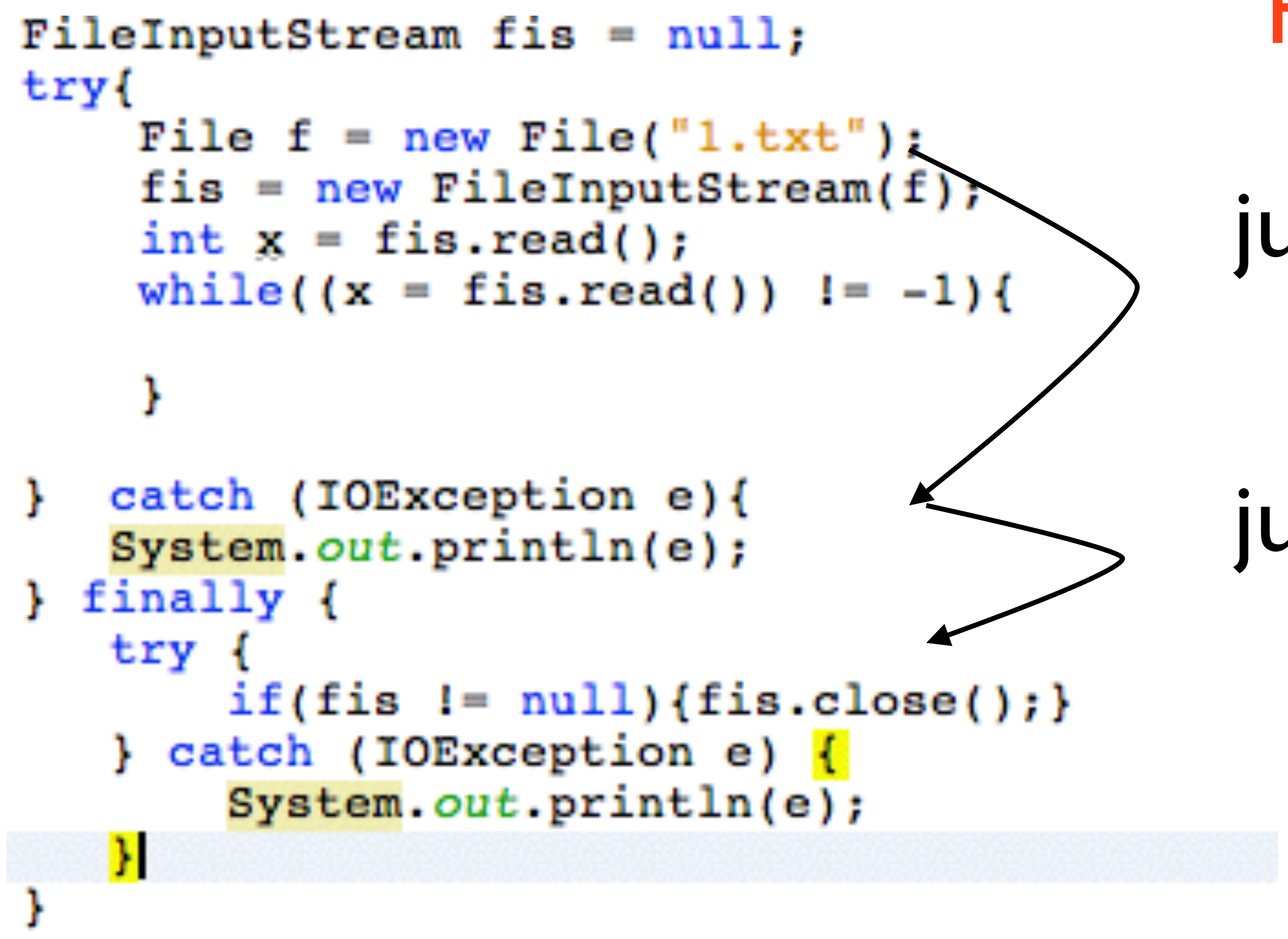
Exception

- what's the keyword when you need to use exception

- try, catch, finally
- throw, throws

Exception

- Describe the flow of the program when the file does not exist and when the file exists



```
FileInputStream fis = null;
try{
    File f = new File("1.txt");
    fis = new FileInputStream(f);
    int x = fis.read();
    while((x = fis.read()) != -1){

    }

}  catch (IOException e){
   System.out.println(e);
} finally {
   try {
       if(fis != null){fis.close();}
   } catch (IOException e) {
       System.out.println(e);
   }
}
```

File Exist

jump into finally

File not Exist

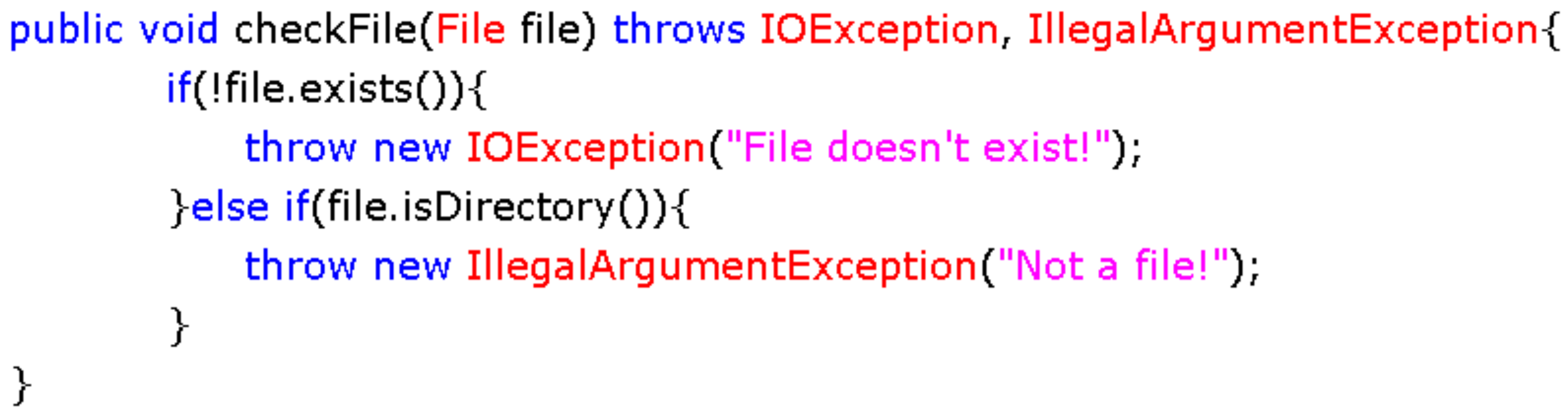
jump into catch

jump into finally

Exception

- throw throws

- throw throws exceptions in method body
- throws defines Exception Specification



```
public void checkFile(File file) throws IOException, IllegalArgumentException{
        if(!file.exists()){
            throw new IOException("File doesn't exist!");
        }else if(file.isDirectory()){
            throw new IllegalArgumentException("Not a file!");
        }
}
```

Exception

- which exceptions I don't have to catch?
  - IOException
  - NullPointerException ✓
  - ArithmeticException ✓
  - StackOverFlowError ✓

I/O

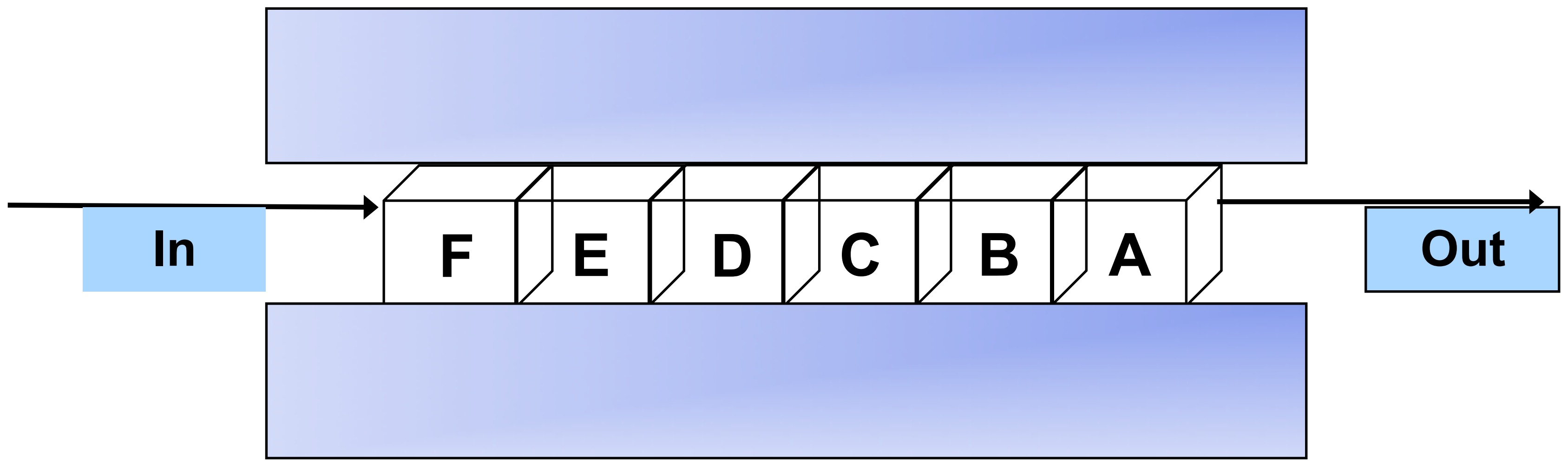
- What’s a File

- java.io.File - “A Path in a file system”
  - File
  - Directory

```
File file = new File("c:/Windows/explorer.exe");
File file = new File("c:/Windows", "explorer.exe");
File file = new File(".");
```

I/O

- What's the character of I/O Stream
  - A sequence of flowing byte / char
  - A channel sending message in FIFO



I/O

- InputStream read()/write()
  - what’s the return type of read()
  - what’s the effective range of read()
  - what’s the value when read method reach the end of file

- int
- 0~255
- -1

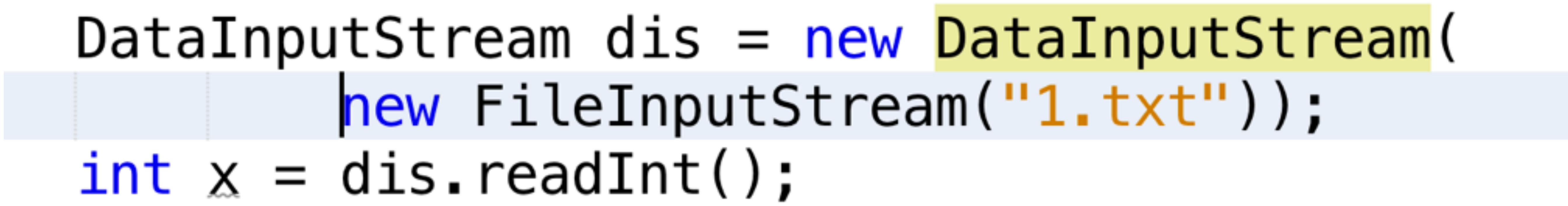
I/O

- Reader read()/write()
  - what’s the return type of read()
  - what’s the effective range of read()
  - what’s the value when read method reach the end of file

- int
- 0~65535
- -1

I/O

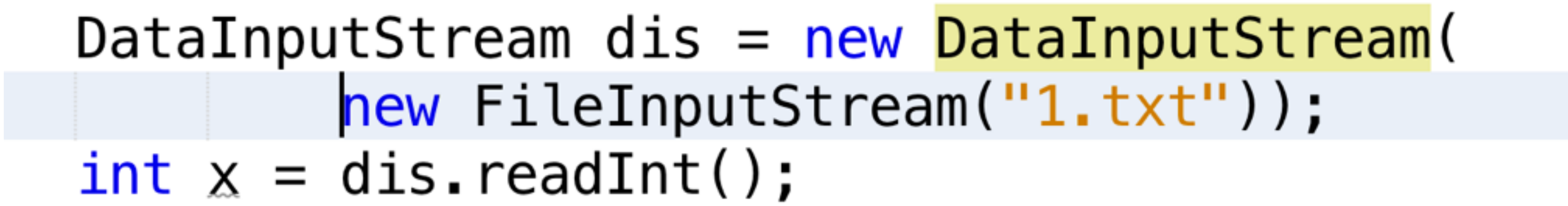
- How read a 4-byte Integer from a file “1.txt”.



```java
DataInputStream dis = new DataInputStream(
        new FileInputStream("1.txt"));
int x = dis.readInt();
```

I/O

- How read a 4-byte Integer from a file “1.txt”.



```
DataInputStream dis = new DataInputStream(
        new FileInputStream("1.txt"));
int x = dis.readInt();
```