Object Oriented Programming

Chapter 10

Ayesha Khatun

Lecturer, Green University of Bangladesh

Exception Handling

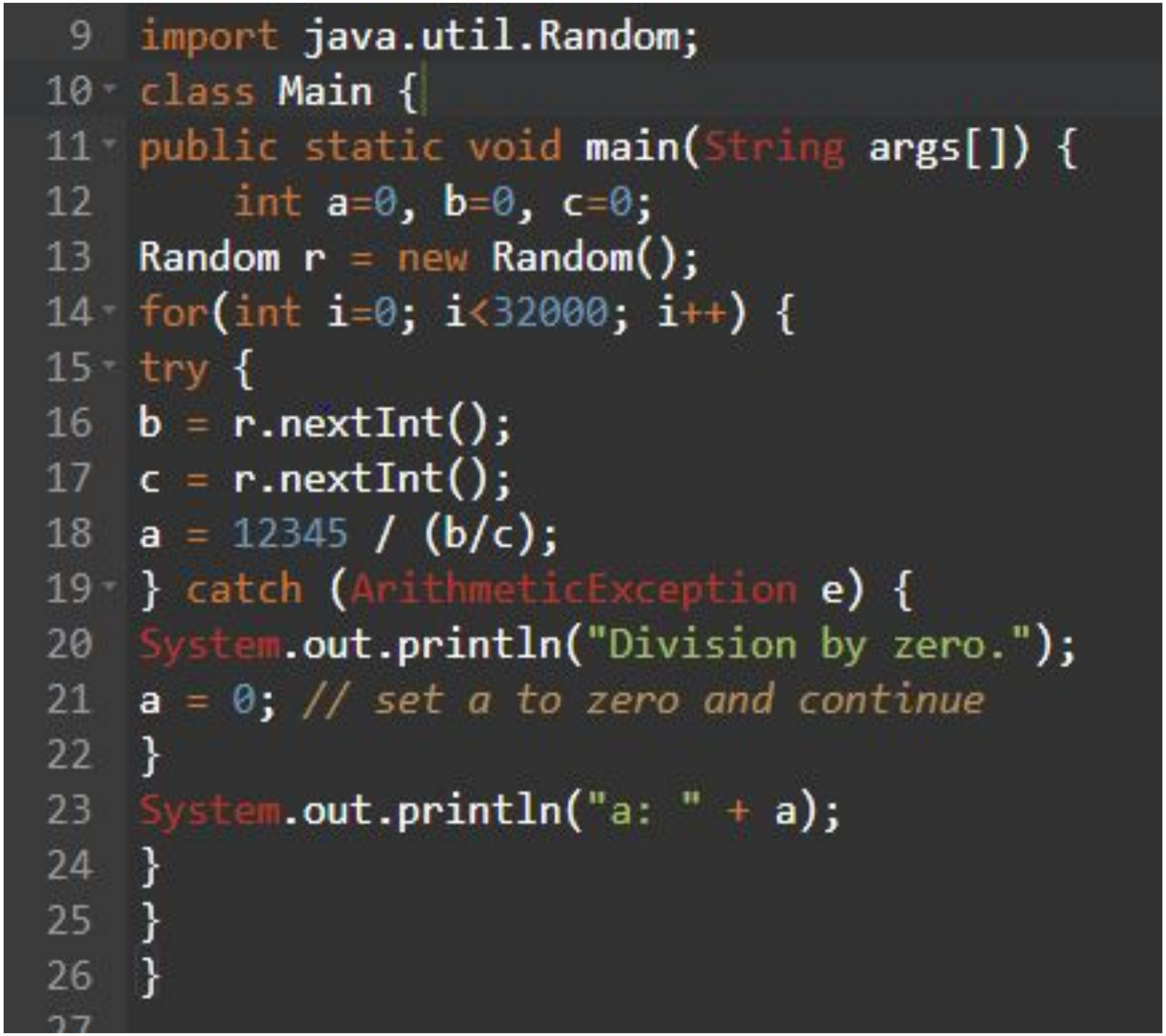
- An *exception is an abnormal* condition that arises in a code sequence at run time.
- Java’s exception handling avoids these problems and, in the process, brings run-time error management into the object oriented world.
  Exception-Handling Fundamentals

Exception-Handling Fundamentals

- try
- catch
- throw
- throws
- finally

Exception-Handling Fundamentals

try {
// block of code to monitor for errors
}
catch (*ExceptionType1 exOb) {*
// exception handler for *ExceptionType1*
}
catch (*ExceptionType2 exOb) {*
// exception handler for *ExceptionType2*
}
// ...
finally {
// block of code to be executed after try block ends
}



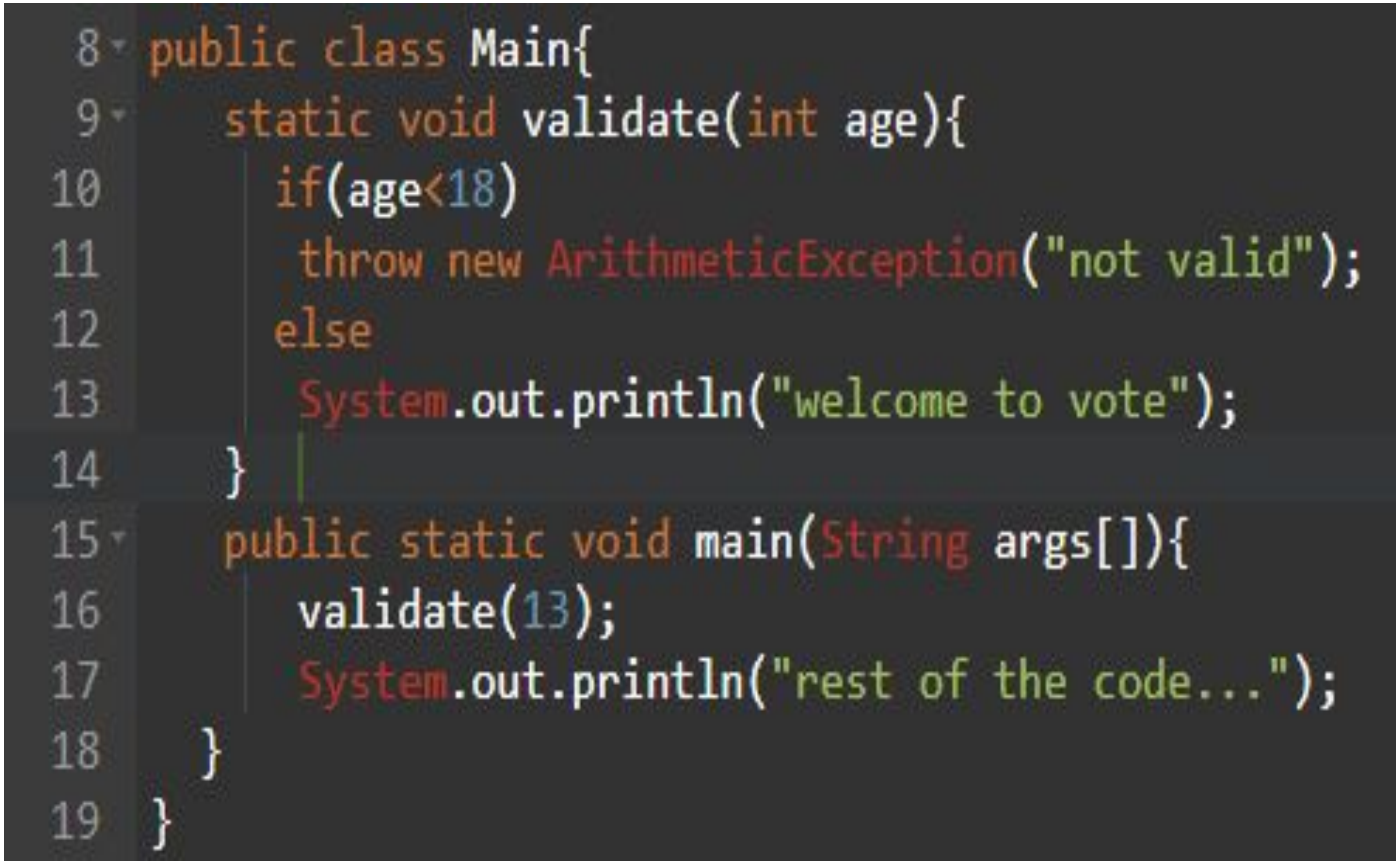
```
import java.util.Random;
class Main {
public static void main(String args[]) {
    int a=0, b=0, c=0;
Random r = new Random();
for(int i=0; i<32000; i++) {
try {
b = r.nextInt();
c = r.nextInt();
a = 12345 / (b/c);
} catch (ArithmeticException e) {
System.out.println("Division by zero.");
a = 0; // set a to zero and continue
}
System.out.println("a: " + a);
}
}
}
```

throw

So far, you have only been catching exceptions that are thrown by the Java run-time system. However, it is possible for your program to throw an exception explicitly, using the **throw** statement. The general form of **throw is shown here:**

throw *ThrowableInstance;*

throw



```
public class Main{
    static void validate(int age){
      if(age<18)
       throw new ArithmeticException("not valid");
      else
       System.out.println("welcome to vote");
    }
    public static void main(String args[]){
       validate(13);
       System.out.println("rest of the code...");
   }
}
```

finally

- **finally creates a block of code that will be executed after a try/catch block has** completed and before the code following the **try/catch block. The finally block will** execute whether or not an exception is thrown.

Exception	**Meaning**
ArithmeticException	Arithmetic error, such as divide-by-zero.
ArrayIndexOutOfBoundsException	Array index is out-of-bounds.
ArrayStoreException	Assignment to an array element of an incompatible type.
ClassCastException	Invalid cast.
EnumConstantNotPresentException	An attempt is made to use an undefined enumeration value.
IllegalArgumentException	Illegal argument used to invoke a method.
IllegalMonitorStateException	Illegal monitor operation, such as waiting on an unlocked thread.
IllegalStateException	Environment or application is in incorrect state.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	Array created with a negative size.
NullPointerException	Invalid use of a null reference.
NumberFormatException	Invalid conversion of a string to a numeric format.
SecurityException	Attempt to violate security.
StringIndexOutOfBounds	Attempt to index outside the bounds of a string.
TypeNotPresentException	Type not found.
UnsupportedOperationException	An unsupported operation was encountered.

TABLE 10-1 Java's Unchecked **RuntimeException** Subclasses Defined in **java.lang**

Thank You!