Exceptions

- *exceptions* are a mechanism for signaling that something unexpected or out of the ordinary has happened (at runtime)
  - an exception is *thrown* at the point where the problem occurs
  - the exception is *caught* (and handled) at the point where the program can do something about the problem
- some kinds of exceptions must be caught
  - not catching these kinds of exceptions is a compiler error
- exceptions that arise from programmer errors (bugs) generally do not require handlers (and are not usually caught)
  - e.g. `IllegalArgumentException`, `ArrayIndexOutOfBoundsException`
  - not catching these kinds of exceptions results in program termination with a stack trace

Throwing Exceptions

Exceptions are thrown at the point where the error is detected.

- the exception type indicates the kind of error
- the detail message provides more specifics about the particular situation

- for violated preconditions

```
if ( precondition is violated ) {
  throw new IllegalArgumentException("detail message");
}
```

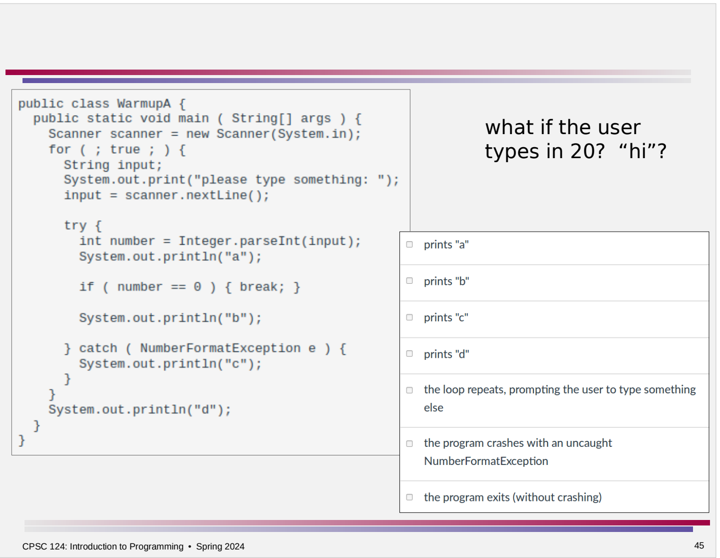
Catching Exceptions

Exceptions are caught at the point where the exceptional situation can be addressed.

- surround code that may throw an exception with `try...catch`
  - this is typically around a subroutine call that can throw an exception rather than the `throw` itself
    - exceptions are typically used when the point of detection is different from where the problem can be addressed
- for exceptions which must be caught, a subroutine that can throw an exception must either
  - contain a try-catch block for that type of exception
  - declare that it throws the exception with `throws` in the subroutine header

Semantics

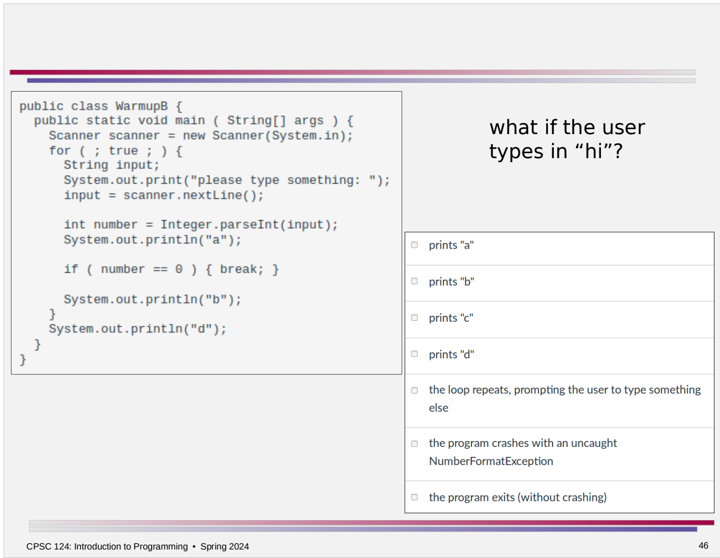
- when an exception is thrown, execution continues with the body of the nearest enclosing catch block and then with whatever follows the catch block
- execution does *not* return to the point immediately after the `throw`



```
public class WarmupA {
  public static void main ( String[] args ) {
    Scanner scanner = new Scanner(System.in);
    for ( ; true ; ) {
      String input;
      System.out.print("please type something: ");
      input = scanner.nextLine();

      try {
        int number = Integer.parseInt(input);
        System.out.println("a");

        if ( number == 0 ) { break; }

        System.out.println("b");

      } catch ( NumberFormatException e ) {
        System.out.println("c");
      }
    }
    System.out.println("d");
  }
}
```

what if the user types in 20? “hi”?

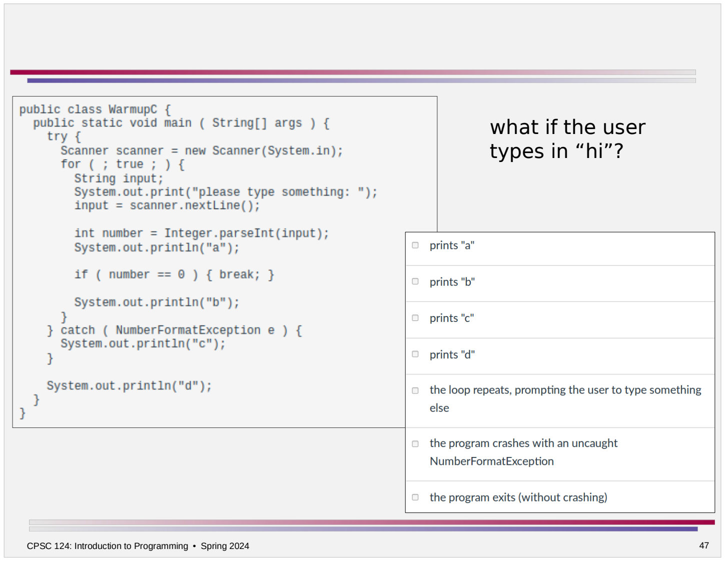
- □ prints "a"
- □ prints "b"
- □ prints "c"
- □ prints "d"
- □ the loop repeats, prompting the user to type something else
- □ the program crashes with an uncaught NumberFormatException
- □ the program exits (without crashing)



```
public class WarmupB {
  public static void main ( String[] args ) {
    Scanner scanner = new Scanner(System.in);
    for ( ; true ; ) {
      String input;
      System.out.print("please type something: ");
      input = scanner.nextLine();

      int number = Integer.parseInt(input);
      System.out.println("a");

      if ( number == 0 ) { break; }

      System.out.println("b");
    }
    System.out.println("d");
  }
}
```

what if the user types in “hi”?

- □ prints "a"
- □ prints "b"
- □ prints "c"
- □ prints "d"
- □ the loop repeats, prompting the user to type something else
- □ the program crashes with an uncaught NumberFormatException
- □ the program exits (without crashing)



```
public class WarmupC {
  public static void main ( String[] args ) {
    try {
      Scanner scanner = new Scanner(System.in);
      for ( ; true ; ) {
        String input;
        System.out.print("please type something: ");
        input = scanner.nextLine();

        int number = Integer.parseInt(input);
        System.out.println("a");

        if ( number == 0 ) { break; }

        System.out.println("b");
      }
    } catch ( NumberFormatException e ) {
      System.out.println("c");
    }

    System.out.println("d");
  }
}
```

what if the user types in “hi”?

- □ prints "a"
- □ prints "b"
- □ prints "c"
- □ prints "d"
- □ the loop repeats, prompting the user to type something else
- □ the program crashes with an uncaught NumberFormatException
- □ the program exits (without crashing)