Exceptions and User Input Validation
The Circle Program, Revisited.

• This is the last version we saw, with looping, and quitting with -1.

```java
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Enter the circle radius, or -1 to quit: ");
            double radius = in.nextDouble();
            if (radius == -1) {
                System.out.printf("Exiting...
");
                break;
            }
            double circumference = 2 * Math.PI * radius;
            double area = Math.PI * Math.pow(radius, 2);
            System.out.printf("Circumference = %.2f.\n", circumference);
            System.out.printf("Area = %.2f.\n\n", area);
        }
    }
}
```
Enter the circle radius, or -1 to quit: 1
Circumference = 6.28.
Area = 3.14.

Enter the circle radius, or -1 to quit: 2.3
Circumference = 14.45.
Area = 16.62.

Enter the circle radius, or -1 to quit: -1
Exiting...

Example Output 1
The Circle Program, Revisited.

- The program crashes when we enter an invalid double number.
- Would be nice to not crash when the input is not valid.
- In general: programs need input validation.

Example Output 2

```
Enter the circle radius, or -1 to quit: 5.2
Exception in thread "main"
    java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextDouble(Scanner.java:2413)
    at example1.main(example1.java:9)
Java Result: 1
```
Strategy for Input Validation

• Read only strings directly from user input.
  – Use only `in.next()`.
  – Do not use `in.nextInt()`, or `in.nextDouble()`. Why?
Strategy for Input Validation

• Read only strings directly from user input.
  – Use only in.next(). This will never lead to a crash.
  – Do not use in.nextInt(), or in.nextDouble(). Why?
    Because they may lead to a crash, if the user enters invalid input.

• If you want to convert string \texttt{str} to a number, use:
  – Integer.parseInt(\texttt{str}) to get an int, or
  – Double.parseDouble(\texttt{str}) to get a double.

• These conversions \textbf{MAY STILL LEAD TO A CRASH}.

• We will see how to avoid such crashes, using \texttt{try ... catch}. 
Converting a String to a Number

• Suppose you have a string \texttt{str}, that you want to convert into a number.

• To convert string \texttt{str} to a number, use:
  – \texttt{Integer.parseInt(str)} to get an int, or
  – \texttt{Double.parseDouble(str)} to get a double.

```java
import java.util.Scanner;

public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("Please enter an integer: ");
        String input = in.next();
        int number = Integer.parseInt(input);
        int square = number * number;
        System.out.printf("%d squared = %d\n", number, square);
    }
}
```

Toy example: computing the square of a number.
import java.util.Scanner;
public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        while (true) {
            System.out.printf("Please enter an integer, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) { // Check if the user wants to quit.
                System.out.printf("Exiting...
");
                break;
            }
            int number = Integer.parseInt(input);
            int square = number * number;
            System.out.printf("%d squared = %d
", number, square);
        }
    }
}

Toy example: computing the square of an integer.
Please enter an integer, or q to quit: 12
12 squared = 144

Please enter an integer, or q to quit: -4
-4 squared = 16

Please enter an integer, or q to quit: q
Exiting...
Things to Note in Previous Example

• We allow the user to quit by typing “q”.
• Why not use -1?
Things to Note in Previous Example

• We allow the user to quit by typing “q”.
• Why not use -1?
• First, a “q” is more intuitive.
• Second (and more important): -1 is a valid number, that the user may enter as normal input.
  – The user may want to compute the square of -1.
Things to Note in Previous Example

• We allow the user to quit by typing “q”. How?
Things to Note in Previous Example

- We allow the user to quit by typing “q”. How?
- We get a string called **input** from the user.
- If **input** is “q”, the program quits.
- Otherwise we convert **input** into a number and continue processing that number.

```java
String input = in.next();
if (input.equals("q"))
{
    System.out.printf("Exiting...\n");
    break;
}
int number = Integer.parseInt(input);
...```
import java.util.Scanner;
public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        while (true) {
            System.out.printf("Please enter a number, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...
");
                break;
            }
            double number = Double.parseDouble(input);
            double square = number * number;
            System.out.printf("%.2f squared = %.2f

", number, square);
        }
    }
}
Double.parseDouble() Example

Please enter a number, or q to quit: 3.2
3.20 squared = 10.24

Please enter a number, or q to quit: -2.1
-2.10 squared = 4.41

Please enter a number, or q to quit: q
Exiting...

Example Output
Double.parseDouble() Crashing

Please enter a number, or q to quit: 5,2

Exception in thread "main" java.lang.NumberFormatException: For input string: "5,2"
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2043)
    at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
    at java.lang.Double.parseDouble(Double.java:538)
    at example1.main(example1.java:17)

Java Result: 1

Example Output

- **Integer.parseInt** and **Double.parseDouble** crash if their argument cannot be converted to an int or double.
- They crash by **throwing an exception**.
- An **exception** is Java’s way of saying “something went wrong”.
Exception Handling (**try** ... **catch**) 

- Suppose that a line of code may make your program crash, by throwing an exception. 
- You can prevent the crash, by **catching the exception**, using **try** ... **catch**. This is called **exception handling**.

```java
try
{
    line_that_mayCause_crash;
}
catch (Exception e)
{
    code for the case where something went wrong.
}
Code for the case where everything went fine.
```
import java.util.Scanner;
public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Please enter an integer, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...
");
                break;
            }
            int number;
            try { // safely convert String input to an integer, catch exceptions.
                number = Integer.parseInt(input);
            } catch (Exception e) {
                System.out.printf("Error: %s is not a valid integer.\n\n", input);
                continue;
            }
            int square = number * number;
            System.out.printf("%d squared = %d
\n", number, square);
        }
    }
}
Integer.parseInt() Example, with try ... catch

Please enter an integer, or q to quit: 5.2
Error: 5.2 is not a valid integer.

Please enter an integer, or q to quit: hello
Error: hello is not a valid integer.

Please enter an integer, or q to quit: -3
-3 squared = 9

Please enter an integer, or q to quit: q
Exiting...

Example Output
Bugs to Avoid (1)

• In the previous example:
  – Variable “number” is declared before the `try ... catch`.
  – Variable “number” is assigned a value in the `try` part of the `try ... catch`.
  – What would go wrong if we did this?

```java
try {
    int number = Integer.parseInt(input);
}
catch (Exception e) {
    System.out.printf("Error: %s is not a valid integer.\n\n", input);
    continue;
}
int square = number * number;
```
Bugs to Avoid (1)

- In the previous example:
  - Variable “number” is declared before the try ... catch.
  - Variable “number” is assigned a value in the try part of the try ... catch.
  - What would go wrong if we did this?

```java
try {
    int number = Integer.parseInt(input);
}
catch (Exception e) {
    System.out.printf("Error: %s is not a valid integer.\n\n", input);
    continue;
}
int square = number * number;
```

Error: number is not defined here.
import java.util.Scanner;

public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Please enter an integer, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...
"); break;
            }
            try // safely convert String input to an integer, catch exceptions.
            {
                int number = Integer.parseInt(input);
            }
            catch (Exception e) {
                System.out.printf("Error: %s is not a valid integer.\n\n", input);
                continue;
            }
            int square = number * number;
            System.out.printf("%d squared = %d\n\n", number, square);
        }
    }
}
```java
import java.util.Scanner;

public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Please enter an integer, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...
");
                break;
            }
            int number;
            try {
                number = Integer.parseInt(input);
            } catch (Exception e) {
                System.out.printf("Error: %s is not a valid integer.\n\n", input);
                continue;
            }
            int square = number * number;
            System.out.printf("%d squared = %d\n\n", number, square);
        }
    }
}```
Bugs to Avoid (2)

• In the previous example:
  – What would go wrong if we deleted the `continue` from the catch part?
import java.util.Scanner;
public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Please enter an integer, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...
");
                break;
            }
            int number;
            try { // safely convert String input to an integer, catch exceptions.
                number = Integer.parseInt(input);
            } catch (Exception e) {
                System.out.printf("Error: %s is not a valid integer.\n\n", input);
            }
            int square = number * number;
            System.out.printf("%d squared = %d\n\n", number, square);
        }}}}
Bugs to Avoid (2)

• In the previous example:
  – What would go wrong if we deleted the `continue` from the catch part?

• The program would proceed with computing the square of `number`.

• However, if `number` fails to be assigned a value, it will have no value there.

• Java will refuse to run this program.
  – Java refuses to run any program that has a chance of using a variable that has not been initialized.

• The `continue` statement reassures Java that `number` will only be used when it has a valid value.
import java.util.Scanner;
public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Please enter an integer, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...\n");
                break;
            }
            int number;
            try // safely convert String input to an integer, catch exceptions.
                number = Integer.parseInt(input);
            catch (Exception e) {
                System.out.printf("Error: %s is not a valid integer.\n\n", input);
                continue;
            }
            int square = number * number;
            System.out.printf("%d squared = %d\n\n", number, square);
        }
    }
}
import java.util.Scanner;
public class compute_square {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Please enter a number, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("Exiting...
");
                break;
            }
            double number;
            try // safely convert String input to a double, catch exceptions.
            {
                number = Double.parseDouble(input);
            }
            catch (Exception e) {
                System.out.printf("Error: %s is not a valid number.\n\n", input);
                continue;
            }
            double square = number * number;
            System.out.printf("%.2f squared = %.2f\n\n", number, square);
        }
    }
}
Double.parseDouble() Example, with try ... catch

Please enter a number, or q to quit: 5.2
5.20 squared = 27.04

Please enter a number, or q to quit: hello
Error: hello is not a valid number.

Please enter a number, or q to quit: 5,2
Error: 5,2 is not a valid number.

Please enter a number, or q to quit: -1
-1.00 squared = 1.00

Please enter a number, or q to quit: q
Exiting...
Strategy for Input Validation, Recap

• Read only strings directly from user input.
  – in.nextInt() and in.nextDouble() may lead to a crash, if the user does not enter a valid number.

• To convert string \texttt{str} to a number, use:
  – \texttt{Integer.parseInt(\texttt{str})} to get an int, or
  – \texttt{Double.parseDouble(\texttt{str})} to get a double.

• These conversions should always be wrapped by \texttt{try \ldots catch}, as shown in the previous examples:

```java
String input = in.next();
double number;
try
{
    number = Double.parseDouble(input);
}
catch (Exception e)
{
    System.out.printf("Error: \%s is not a valid number.\n\n", input);
    continue;
}
```
The Final Circles Program

• We are now ready for the final version of the Circles program.

• One final version is shown on the next slide. It includes:
  – A main loop, so that the user can perform as many calculations as she or he wants.
  – Input validation, making sure that the input is a valid number.
  – Quitting with "q".

• A longer final version is shown on the course website (under example programs for this lecture).
  – Makes sure that the radius is positive, prints an error message otherwise.
import java.util.Scanner;

public class final_circles_program {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.printf("Enter the circle radius, or q to quit: ");
            String input = in.next();
            if (input.equals("q")) {
                System.out.printf("\nExiting...
");
                break;
            }
            double radius;
            try {
                radius = Double.parseDouble(input);
            } catch (Exception e) {
                System.out.printf("Error: %s is not a valid radius.\n\n", input);
                continue;
            }
            double circumference = 2 * Math.PI * radius;
            double area = Math.PI * Math.pow(radius, 2);
            System.out.printf("Circumference = %.2f.\n\n", circumference);
            System.out.printf("Area = %.2f.\n\n", area);
        }
    }
}

The final Circles program:
• A main loop.
• Input validation.
• Quitting with "q".
Enter the circle radius, or q to quit: hello
Error: hello is not a valid radius.

Enter the circle radius, or q to quit: 1
Circumference = 6.28.
Area = 3.14.

Enter the circle radius, or q to quit: 2.3
Circumference = 14.45.
Area = 16.62.

Enter the circle radius, or q to quit: q

Exiting...