Learn more about our research, discover data science, and find other great resources at:

http://www.dataminingapps.com

Chapter 6
Handling Exceptions and Debugging

Overview

- Recognizing error types
- Exceptions
- Debugging Your Applications
- Testing Your Applications

Recognizing Error Types

- Identifying syntax errors
- Identifying runtime errors
- Identifying logical errors

Identifying Syntax Errors

- Misspelled class, variable, or method names
- Misspelled keywords
- Missing semicolons
- Missing return type for methods
- Out of place or mismatched parentheses and brackets
- Undeclared or uninitialized variables
- Incorrect format of loops, methods, or other structures

Identifying Syntax Errors

```
public class errors {

public static vod main(String[] args) {
age = 30;
int retirementFund = 10000;
int yearsInRetirement = 0;
String name = "David Johnson",
for (int i = age; <= 65; ++){
   recalculate(retirementFund,0.1);
}
int monthlyPension = retirementFund/yearsInRetirement/12
System.out.printline(name + " will have $" + monthlyPension
+ " per month for retirement."];
}

public static recalculate(fundAmount, rate){
fundAmount = fundAmount*(1+rate);
}}}
```

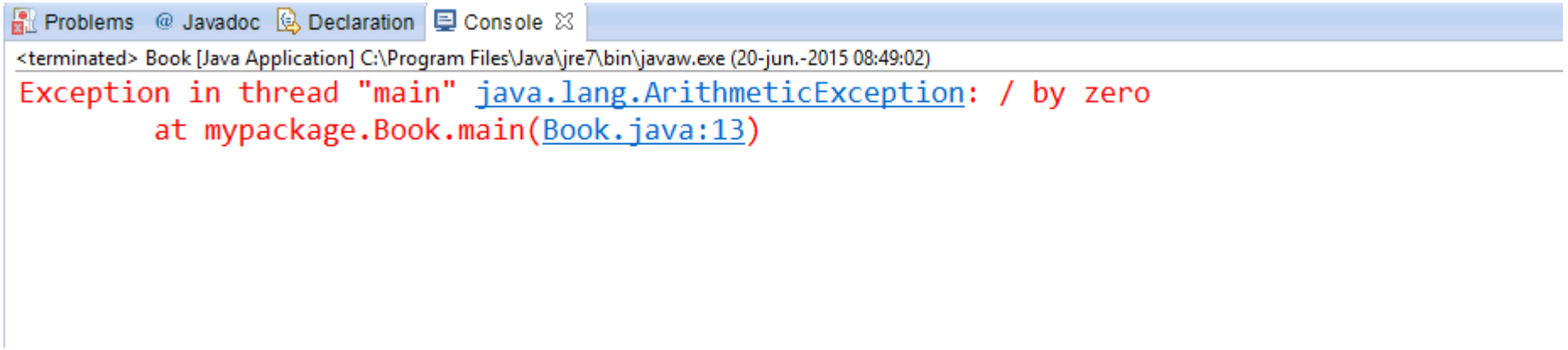
Identifying Syntax Errors

```
public class errors {

public static void main(String[] args) {
int age = 30;
int retirementFund = 10000;
int yearsInRetirement = 0;
String name = "David Johnson“;
for (int i = age; i <= 65; i++){
   recalculate(retirementFund,0.1);
}
int monthlyPension = retirementFund/yearsInRetirement/12;
System.out.println(name + " will have $" + monthlyPension
+ " per month for retirement.“);
}

public static void recalculate(double fundAmount, double rate){
fundAmount = fundAmount*(1+rate);
}
}
```

Identifying Runtime Errors



Line 13: `int monthlyPension = retirementFund/yearsInRetirement/12;`

Remember: `int yearsInRetirement = 0;`

Identifying Logical Errors

```
public class errors {

public static void main(String[] args) {
int age = 30;
int retirementFund = 10000;
int yearsInRetirement = 20;
String name = "David Johnson“;
for (int i = age; i <= 65; i++){
   recalculate(retirementFund,0.1);
}
int monthlyPension = retirementFund/yearsInRetirement/12;
System.out.println(name + " will have $" + monthlyPension
+ " per month for retirement.“);
}

public static void recalculate(double fundAmount, double rate){
fundAmount = fundAmount*(1+rate);
}}
```

Output:

David Johnson will have $41 per month for retirement.

Identifying Logical Errors

```
public class errors {

public static void main(String[] args) {
int age = 30;
double retirementFund = 10000;
int yearsInRetirement = 20;
String name = "David Johnson“;
for (int i = age; i <= 65; i++){
   retirementFund = recalculate(retirementFund,0.1);
}
double monthlyPension = retirementFund/yearsInRetirement/12;
System.out.println(name + " will have $" + monthlyPension
+ " per month for retirement.“);
}

public static double recalculate(double fundAmount, double rate){
return fundAmount*(1+rate);
}}
```

Output:

David Johnson will have $1288.0283555362819 per month for retirement.

Identifying Logical Errors

```
import java.math.BigDecimal;
public class Errors {

public static void main(String[] args) {
int age = 30;
BigDecimal retirementFund = new BigDecimal("10000.00");
// set the scale to 2 decimal points
// and the rounding to round up when the next digit is >= 5

retirementFund.setScale(2,BigDecimal.ROUND_HALF_UP);
BigDecimal yearsInRetirement = new BigDecimal("20.00");
String name = "David Johnson";

for (int i = age; i <= 65; i++){
   retirementFund = recalculate(retirementFund,new
                    BigDecimal("0.10"));
}

BigDecimal monthlyPension =
retirementFund.divide(yearsInRetirement.multiply(new
BigDecimal("12.00")),2, BigDecimal.ROUND_HALF_UP);

System.out.println(name + " will have $" + monthlyPension+ "
per month for retirement.");
}
…
```

```
public static BigDecimal recalculate(BigDecimal
fundAmount, BigDecimal rate){
// use BigDecimal methods for arithmetic
// operations
return fundAmount.multiply(rate.add(new
BigDecimal("1.00")));
}
}
```

Output:

David Johnson will have $1288.03 per month for retirement.

Exceptions

- Events that disrupt the execution of a program
- Common exceptions
  - NullPointerException: accessing an object that doesn't exist/has not been intialized
  - IndexOutOfBoundsException: try to access an element outside the limits of an indexed object (e.g. array)
  - StackOverFlowError: too many parameters and/or local variables created due to e.g. recursion
  - OutOfMemoryError: too many objects created due to e.g. infinite loop

Exceptions

```
public class ExceptionExamples {

public static void main(String[] args) {
Person employee;
printPerson(employee);
}
public static void printPerson(Person myPerson){
System.out.println(myPerson.name + " is " + myPerson.age + " years
old.");
}}

class Person{
String name;
int age;
Person (){
}}
```

Exceptions

```
public class ExceptionExamples {

public static void main(String[] args) {
Person employee;
printPerson(employee);
}
public static void printPerson(Person myPerson){
System.out.println(myPerson.name + " is " + myPerson.age + " years
old.");
}}

class Person{
String name;
int age;
Person (){
}}
```

Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The local variable employee may not have been initialized.

Exceptions

```
public class ExceptionExamples {

public static void main(String[] args) {
Person employee= new Person();
printPerson(employee);
}
public static void printPerson(Person myPerson){
System.out.println(myPerson.name + " is " + myPerson.age + " years
old.");
}}

class Person{
String name;
int age;
Person (){
}}
```

Output:

null is 0 years old.

Exceptions

```
public class ExceptionExamples {
public static void main(String[] args)
{
Person employee = new Person();
printPerson(employee);
}
public static void printPerson(Person
myPerson){
System.out.println(myPerson.name + "
is " + myPerson.age + " years old and
works as a " + myPerson.job.jobName);
}
}
…
```

```
class Person{
String name;
int age;
JobType job;
Person (){
}
}
class JobType{
String jobName;
int salaryBand;
JobType (){
}
}
```

Exceptions

```
public class ExceptionExamples {
public static void main(String[] args)
{
Person employee = new Person();
printPerson(employee);
}
public static void printPerson(Person
myPerson){
System.out.println(myPerson.name + "
is " + myPerson.age + " years old and
works as a " + myPerson.job.JobName);
}
}
…
```

```
class Person{
String name;
int age;
JobType job;
Person (){
}
}
class JobType{
String JobName;
int salaryBand;
JobType (){
}
}
```

Output:

Exception in thread "main" java.lang.NullPointerException

Exceptions

```
public class ExceptionExamples {

public static void main(String[] args) {
JobType manager = new JobType("Manager", 6);
Person employee = new Person("Bob Little", 47,
manager);
printPerson(employee);
}

public static void printPerson(Person myPerson){
System.out.println(myPerson.name + " is " +
myPerson.age +
" years old and works as a " +
myPerson.job.JobName);
}
}
…
```

Output:

Bob Little is 47 years old and works as a Manager

```
class Person{
String name;
int age;
JobType job;

Person (String name, int age, JobType job){
  this.name = name;
  this.age = age;
  this.job = job;
}
}
class JobType{
String JobName;
int salaryBand;

JobType (String name, int band){
  JobName = name;
  salaryBand = band;
}}
```

Exceptions

```
public class IndexExceptionExample {

public static void main(String[] args) {
int[] hoursWorked = {7,8,7,9,5};
int totalHours = 0;

for (int i = 0; i <= hoursWorked.length; i++){
totalHours += hoursWorked[i];
}
System.out.println("Total Hours = " + totalHours);
}
}
```

Exceptions

```
public class IndexExceptionExample {

public static void main(String[] args) {
int[] hoursWorked = {7,8,7,9,5};
int totalHours = 0;

for (int i = 0; i <= hoursWorked.length; i++){
totalHours += hoursWorked[i];
}
System.out.println("Total Hours = " + totalHours);
}
}
```

Output:

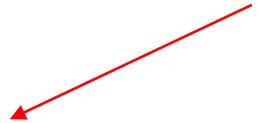
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException

Exceptions

```
public class IndexExceptionExample {

public static void main(String[] args) {
int[] hoursWorked = {7,8,7,9,5};
int totalHours = 0;

for (int i = 0; i < hoursWorked.length; i++){
totalHours += hoursWorked[i];
}
System.out.println("Total Hours = " + totalHours);
}
}
```



Exceptions

```
import java.util.ArrayList;

public class EndlessLoop {

static ArrayList<String> myStrings = new ArrayList<String>();

public static void main(String[] args) {
for (int i = 0; i >= 0; i++) {
myStrings.add("String number: " + i);
}
}
}
```

Exceptions

```
import java.util.ArrayList;

public class EndlessLoop {

static ArrayList<String> myStrings = new ArrayList<String>();

public static void main(String[] args) {
for (int i = 0; i >= 0; i++) {
myStrings.add("String number: " + i);
}
}
}
```

Output:

Exception in thread "main“ java.lang.OutOfMemoryError

Exceptions

```
public class EndlessMethodCall {

public static void main(String[] args) {
printMe();
}

public static void displayMe(){
printMe();
}

public static void printMe(){
displayMe();
}
}
```

Exceptions

```
public class EndlessMethodCall {

public static void main(String[] args) {
printMe();
}

public static void displayMe(){
printMe();
}

public static void printMe(){
displayMe();
}
}
```

Output:
Exception in thread "main" java.lang.StackOverflowError

Catching Exceptions

- `try/catch` block
- Syntax:

```
try {
 // execute some statements
} catch (Exception exc){
 // statements to handle the exception
} finally {
 // no matter what, do this
}
```

Catching Exceptions

```
public class Error {

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

try{
int ratio= numerator/denominator;
System.out.println(ratio);
} catch (ArithmeticException ae){
   System.out.println("Denominator should not be 0!");}

}}
```

Output:

Denominator should not be 0!

Catching Exceptions

```
public class Error {

public static void main(String[] args) {
double numerator=10;
double denominator=0;

try{
  double ratio= numerator/denominator;
  System.out.println(ratio);
} catch (ArithmeticException ae){
  System.out.println("Denominator should not be 0.");}
}}
```

Output:

Infinity

Catching Exceptions

```
public class Error {

public static void main(String[] args) {
double numerator=10;
double denominator=2;

try{
if (denominator == 0) {throw new ArithmeticException();};
double ratio= numerator/denominator;
System.out.println(ratio);
} catch (ArithmeticException  ae){
   System.out.println("Denominator should not be 0!");}
}
}
```

Output:

Denominator should not be 0!

Catching Exceptions

```
public class Error {

public static void main(String[] args) {
double numerator=10;
double denominator=2;

try{
if (denominator == 0) {throw new ArithmeticException();};

double ratio= numerator/denominator;
System.out.println(ratio);
} catch (ArithmeticException  ae){
System.out.println("Denominator should not be 0.");}

finally{
System.out.println("Finally was reached!");}
}}
```

Output:

5.0

Finally was reached!

Catching Exceptions

```
public class Error {

public static void main(String[] args) {
double numerator=10;
double denominator=0;

try{
if (denominator == 0) {throw new ArithmeticException();};

double ratio= numerator/denominator;
System.out.println(ratio);
} catch (ArithmeticException  ae){
System.out.println("Denominator should not be 0.");}


finally{
System.out.println("Finally was reached!");}
}}
```

Output:

Denominator should not be 0.
Finally was reached!

Catching exceptions

```
import java.text.DecimalFormat;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Error {
static Scanner scan = new Scanner(System.in);

public static void main(String[] args) {
try {
  System.out.print("Enter the loan amount: ");
  double principle = scan.nextDouble();
  System.out.print("Enter the interest rate: ");
  double rate = scan.nextDouble();
  System.out.print("Enter the loan term (in years): ");
  double years = scan.nextInt();
  double interest = principle*rate*years;
  double total = principle + interest;
  double payment = total/years/12;
  DecimalFormat df = new DecimalFormat ("0.##");
  System.out.println("Monthly payment: $“ + df.format(payment));
} catch (InputMismatchException exc){
…
System.out.println("Please provide correct
input!");
} finally {
scan.close();
}}}
```

Input:

Enter the loan amount: Bart

Output:

Please provide correct input!

Catching exceptions

```
import java.text.DecimalFormat;
import java.util.Scanner;

public class Error {

public static void main(String[] args) {
try (Scanner scan = new Scanner(System.in)){
  System.out.print("Enter the loan amount: ");
  double principle = scan.nextDouble();
  System.out.print("Enter the interest rate: ");
  double rate = scan.nextDouble();
  System.out.print("Enter the loan term (in years): ");
  double years = scan.nextInt();
  double interest = principle*rate*years;
  double total = principle + interest;
  double payment = total/years/12;
  DecimalFormat df = new DecimalFormat ("0.##");
  System.out.println("Monthly payment: $" + df.format(payment));
} catch (Exception exc){
  System.out.println(exc);
}}}
```

Catching exceptions

```
import java.text.DecimalFormat;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Error {
public static void main(String[] args) {
try {
double[] userValues = scanValues();
double payment =
(userValues[0]+userValues[0]*userValues[1]*userValues[2])/
userValues[2]/12;
DecimalFormat df = new DecimalFormat("0.##");
System.out.println("Monthly payment: $" +
df.format(payment));
} catch (InputMismatchException ime) {
System.out.println("You must enter double values! ");
 System.exit(0);
} catch (ArithmeticException ae) {
System.out.println("Arithmetic error! ");
System.exit(0);
} catch (IndexOutOfBoundsException ioob) {
System.out.println("Three doubles are required! ");
System.exit(0);
}}
```

```
public static double[] scanValues() throws
InputMismatchException {
  double[] values = new double[3];
  Scanner scan = new Scanner(System.in);
  try {
   System.out.print("Enter loan amount: ");
   values[0] = scan.nextDouble();
   System.out.print("Enter interest rate: ");
   values[1] = scan.nextDouble();
   System.out.print("Enter loan term: ");
   values[2] = scan.nextInt();
} finally {
  scan.close();
}
return values;
}}
```

Input:

Enter the loan amount: Bart

Output:

You must enter double values!

Conclusions

- Recognizing error types
- Exceptions
- Debugging Your Applications
- Testing Your Applications