2019 Q1 Programming Summary

Running a Java program: (1) type the text with at least 1 class (2) save file with name class (3) compile file (checks for errors) (4) run the class file.
For naming things, use CamelCase(start each new word in a name with an uppercase letter)
Types of rules: (1) syntax (grammar) (2) semantic (meaning) (3) conventions (wise advice)

Topic	**How to write it in Java**	**What it means**
How to make a program	public class <nameClass> { <statements> }	This is the class declaration and is at the beginning of every source code, it contains the whole program.
	void <method>() { <statements> }	Method declaration, it contains the activity of the program.
	System.out.println("<text>");	Gives output and newline statement
	public static void main(String[] args) { new <class>().<method>(); }	This is at the end of your program. The first part marks the start-up code and the second part creates an active object from the class and starts the method.
Numeric/ primitive data types	int <name variable>;	This creates a variable that is a integral number. Use CamelCase but start with a lower case letter. Values: …, -1, 0, 1, … . Operations: +, -, *, /, %. E.g. (17+4)/2 gives 10. Equality by x==y, inequality by x!=y
	Double <name variable>;	This creates a variable that is a floating point number. Values: … , -0.5, …, 0.7, … . (in)equal: see int. Operations: +, -, *, / E.g. (17.0+4.0)/2.0 gives 1.5
Arithmeric operators (on numeric types)	5 * 2 gives 10 Int: 5 / 2 gives 2. D: 5.0 / 2 gives 2.5 1 + 1 gives 2. "1" + "1" gives "11" 5 – 2 gives 3. – (5 - 2) gives -3 10 % 5 gives 0. 7 % 5 gives 2 Math.pow(x,y)	Multiplication Integer division (ordinary division) Arithmetic addition (string concatenation) Subtraction (minus) Remainder or modulo (rest bij deling) Machtsverheffen (x^y)
Textual data types	char <name variable>;	This creates a variable that is a single character. Values: 'a', '0', '?', … . No operations!
	String <name variable>;	This creates a variable that has sequences of characters (also "a"!). Values: "seven", "a", … Operations: +, … E.g. "seven" + "teen" gives "seventeen". Equal identity: x==y. To compare strings, use "<String>".equals("..."). To look at the alphabetical ordering, use "<String1>".compareTo("<String2>"). 0: same place, -1: String1 is closer to beginning than String2, 1: vice versa. "<String>".length() gives the number of characters. "<String>"substring: gives subsequence by position (e.g."Building".substring(1,4) → "uil") "<String>".indexOf("…"):gives position of substring (e.g. "Building".indexOf("uil") → 1)
Data type Boolean	boolean <name variable>;	This creates a variables that can evaluate to true or false, depending on the input. Values: true, false Operations: && (logical and), || (inclusive or), ! (logical not), ^ (exclusive or), == (equality)

2019 Q1 Programming Summary

Comparison operators	5 +7 == 24 / 2 <string1>.equals(<string2>)	Comparing all primitive types (int, double, Boolean) gives a true or false, it's a Boolean value. Comparing String types (DON'T USE ==)

Scanner Input	import java.util.Scanner;
	class <name> {
	Double amount;
	This: Scanner = new
	Scanner(System.in);
	Public void <method>() {
	amount = scanner.nextDouble(); }
	Or like this: Scanner;
	Public void <method>() {
	scanner = new Scanner(System.in);
	amount = scanner.nextDouble; }
		The user can insert values which can be used in the program by applying a scanner. Here, you can see all the necessary parts of using a scanner.
		When no input is available, the scanner waits for input.
		"scanner" can also be replaced by another word.
		Variants of "scanner.nextDouble()" (which reads the next Double & stores it) are "scanner.nextInt()" (for integers), "scanner.next()" (for Strings/a word), "scanner.nextLine()" (for lines (whole line=String)).
		It is also possible to "read" whether there even is a Int or a Double with "scanner.hasNext()" (e.g. scanner.hasNextDouble() checks for Doubles). These statements will return "true" or "false.

If-construct	if (condition) { ... then-part } **else {** **... else-part** **}**	The if-construct is used to execute statements only when the condition is true, uses Boolean expressions. Here you can see the if-else construct. If you leave out the part in bold, you have a if-construct with one branch only.

	if (A) {
	... executed if A is true
	} else if (B) {
	... executed if B is true and A is not true
	} else if (C) {
	... executed if C is true and neither A nor B are true
	} else {
	... executed if A nor B nor C is true
	}
		Here you can see the if-else if construct.

	if (A1) {
	if (B1) {
	... A1 true and B1 true
	} else {
	... A1 true and B1 false
	}
	} else {
	if (B2) {
	... A1 false and B2 true
	} else {
	... A1 false and B2 false
	} }
		Here you can see the usage of nesting. This is hard to manage so it is better not to use it!
		If you do, try to figure out what each branch means: this way you can avoid errors.

2019 Q1 Programming Summary

Comments	// <comment> /* <comment> */ /** <comment> * <comment> */	Comments are text that won't be read by the computer. Comments are mainly used to help other developers understand the program.
Interpolation	String.format("x=%d, y=%d, z=%d", x, y, z) gives "x=-1, y=10, z=-3" (depending on values of x, y, z) String.format("%s is better than %s", "PSV", "Ajax") gives "PSV is better than Ajax" String.format("<.2f>", Math.PI) → <3.14> String.format("<6.2f>", Math.PI) → < 3.14>	You can use interpolation for convenient printing. %s: applies to any type, output: String value %d:applies to integer, output: decimal representation %f: applies to float/double, output: decim. repres. %e: applies to float/double, output: scientific notation %nX: applies to X any specifier, effect: width use n positions %.nX: applies to X a floating point specifier, effect: use n decimals %-nX: applies to X any specifier, effect: width n, left aligned %n.mX: applies to X a floating point specifier, effect: m decimals, total width n

While-construct	while (<boolean expression>) {
	<statements>
	}
		The while-construct executes some statements more than one. The loop-body (statements) is repeated while guard (a Boolean expression) is true; guard is inspected before each execution.

	int n;
	n = 0;
	while (n<3) {
	System.out.println("Hello");
	n = n + 1;
	} }
		Here is a while-construct with a fixed number of iterations (chosen by programmer). This can be seen by n = 0.

	int times;
	int n;
	System.out.println("Choose number of times");
	times = sc.nextInt();
	n = 0;
	while (n < times) {
	System.out.println("Hello");
	n = n + 1; } }
		Here is a while-construct with a chosen number of iterations (chosen by user). This can be seen by the usage of a scanner.

	String choice;
	System.out.println("For Hello type h, type anything else to stop");
	choice = sc.next();
	while ("h".equals(choice)) {
	System.out.println("Hello");
	choice = sc.next();
	} }
		Here is a while-construct that repeats while the user wants to. This can be seen in the guard: "h".equals(choice).

2019 Q1 Programming Summary

	Double balance;
	Double rate = 5;
	int n = 0;
	System.out.println(“Fill in balance”);
	balance = sc.nextDouble();
	while (balance < 1000000) {
	System.out.println(“balance after
	”+n+” years: ”+balance);
	n = n + 1;
	balance=balance+rate*balance/100
	} System.out.println(“Millionaire!” }
		Here is a while-construct that repeats until result. This can be seen in the guard: balance < 1000000.

	Not possible!	This is the while-construct that repeats until told to stop.

For-loop	for (<initialization> ; <guard> ;
	<progress>) {
	<body>
	}
		The for-loop is used for when you want to do something for each member of a set of values and when you know the number of iterations (times the loop-body is executed) on beforehand. Initialization: first assignment to counter or other variables (int i=0). Progress: often increasing a counter(i=i+1).Guard: Boolean expression.

Do-while loop	do { <statements> } while (<guard>);	The do-while loop is used when you have to do something at least once. First, it executes the code and then it checks whether the guard is true. Remark: i++ and i+=1 are short for i=i+1 (also possible for *=, -+, %=, etc)

For-each loop	for (<type> <variable> : <collection>)
	{
	<statements>
	}
		Often each element of an array has to be visited: the for-each loop gives exactly this without need for the index variable and it works for all kinds of collections. Type: a member (element) type of collection. Collection: array or other collection with members of type <type>. Variable: variable that will contain the members of the collection.

Arrays	int[] students;
	students = new int[3];
	students[0] = 7;
	students[1] = 5;
	students[2] = 3;
	
	System.out.println(student.length);
		An array is a sequence of numbered variables. The number between [] is the index. All arrays count from 0. Declaration and creation are separate but can be combined (int[] students = new int[3];).
		Trying to access an array element that doesn’t exist is a serious error!
		This outputs 3. (NOT SURE)

ArrayList		An ArrayList is an array of flexible length (grows when more elements are added, shrinks when elements are deleted). size() is used to determine the number of members of an ArrayList.

2019 Q1 Programming Summary

ArrayList vs Array	ArrayList<String> a;
	a = new ArrayList<String>();
	a = new ArrayList<>();
	a.add(“aap”);
	a.add(i,”jet”);
	a.set(0,”AAP”);
	s=a.get(0)+a.get(1)
	a.remove(i);
	for (String s:a){…s…}
	for(int i=0; i<a.size(); i++) { …a.get(i)… }
	
	a.size()
	a.equals(b)
	
	a.toString()
	System.out.println(a)
		String[] a;
		a = newString[…];
		
		-
		-
		a[0] = “AAP”;
		a = a[0]+a[1]
		-
		for(String s:a){…s…}
		for(int i=0; i<a.size(); i++) {…a.get(i)…}
		a.length
		Array.equals (a,b)
		Árrays. toString(a)
		System.out. println(Arrays. toString(a))
			Declaration
			Creation
			
			Appending (to the end of the list)
			Insertion (elements from i on are shifted to the right
			Change
			Inspection
			Deletion
			Visit all elements
			
			Number of elements
			Equality of contents
			
			Printen etc.

Multi-dimensional arrays	String[][] table;
	table = new String[3][4];
	table[0][0] = "good";
	table[1][3] = "exc";
	for (int c=0; c < table[2].length; c++) {
	table[2][c] = "avg";
	}
	for (int r=0; r < table.length; r++) {
	table[r][1] = "bad";
	}
			Multidimensional arrays are used to model grids, matrices, etc. The example on the left gives a table with 3 rows and 4 columns; the cell in the first column and the first row has as output “good”.
			Two-dimensional arrays can show pixels (x,y) and matrices (row, column). It doesn’t matter which system, as long as you stick to it.

Methods	void MethodName() {
	//code in method
	}
	MethodName(); //calling the method
			You can “call” a method. Then the code will be executed. There are 4 method types:
			- simple: void <name>() { <body> }
			- parameters: void<name>(<type> <var>, …, <type> <var>) { <body>)
			- return: <type> <name>() { <body> return <val>; }
			- combination of parameters and return

Variables			Three kinds: - instance variable: scope is all methods in class - parameter: scope is in one method, is in the brackets of the method (e.g. name(HERE)), input of method – local variable: scope is in one method/less, output “

Objects	class Garden {
	double width; double length; }
	Garden g = new Garden();
	g.width = 10; g.length = 12;
			You can make a class
			
			And then create an object using that class (it becomes an instance of a class)

2019 Q1 Programming Summary

Constructor	class Garden {
	double length; double width;
	Pond pond;
	Garden(double len, double wid, Pond p) { this.length = len;
	this.width = wid; this.pond = p;
	} }
		Is very much like a method but different: Constructor:
		- Has no return type, not even void
		- Has the same name as the class
		- Can only be called at construction
		- Cannot be inherited

Alias	Garden g = new Garden(5,3); Garden h = g; g.setLength(10);	Two part of the code share an object Changes to g are sensed also by h So: g.getArea() and h.getArea() both give 30
Inheritance	Class MarketGarden extends Garden { <body> } TaxInspection { Garden[] gardens; double totalTax(); }	Existing instance variables and methods of Garden are "inherited" (automatically copied). So MarketGarden is a Garden. TaxInspection has Garden. New class is a subclass of existing class (superclass). Methods can be redefined/overriding. Constructors are not inherited and have to be repeated using super. Overriding can be done by defining a method with the same signature (name and parameter types). Mark this overriding method with @Override. Use super to reuse code form superclass instead of copying (e.g. super.calcTax();) A Java class can extend (inherit from) *only one class*. A Java class can implement *several types* (interfaces and classes). A Java object can have several types. An interface extends one or more interfaces. An interface can not extend/implement a class.

Graphics	class FruitPanel extends JPanel {
	@Override
	public void paintComponent(Graphics g) {
	super.paintComponent(g);
	g.drawString("Eet meer Freud, dan bleibt u Jung!", 40, 100);
	g.setColor(Color.orange);
	g.fillOval(40, 20, 50, 50);
	}
		There are drawing methods in Graphics. These can be drawn on a JPanel.
		For example:
		draw/fillOval(top_left_x, top_left_y, width, height)
		draw/fillRect, draw/fillRoundRect, drawFill3DRect
		drawString(string, btm_left_x, btm_left_y)
		drawLine
		drawImage
		draw/fillPolygon
		draw/fillArc

Access Modifiers	public //no limitations protected //visible in package and subclasses <none> //visible in same package private //visible only in same class final //prevents change of a variable (no inheritance) or class static //thing belongs to class, can't use "this", acces is via name of class, stores data for the whole class	The goal of this is to decrease dependency between classes. This can be done by restricting visibility of methods and variables.

2019 Q1 Programming Summary

Event mechanism	panel = new JPanel() //create event source
	class MyListener implements MouseListener {
	public void mousePressed (MouseEvent evt) { evt.getX() … }
	public void mouseReleased (MouseEvent evt) { evt.getX() … }
	
	myListener = new MyListener(...);
	panel.addMouseListener(myListener);
		An event mechanism happens when the code reacts to event sources. This could happen to a mouse:
		//make a class that defines the listener
		
		//define the event method(s) in that class
		This can be done for mousePressed, mouseReleased, mouseClicked (when pressing and releasing without moving the mouse), mouseEntered and mouseExited.
		//create an object of that class
		//register the object with the source

	See Mouse but then : button = new JButton("Click me"); class MyListener implements ActionListener { }	Buttons (JButton) are objects on the screen with text/picture that when clicked trigger a reaction. It has the same steps as mouse.

	class C implements ActionListener {
	public void actionPerformed(ActionEvent e) {
	<handler code>
	}
	C c = new C();
	addActionListener(c);
		Many interfaces have only one method, one which you can see on the left.
		"ActionEvent e" is the argument
		The handler code is the behaviour.

	JButton – when clicked
	JComboBox – when item selected
	JTextField – when return is pressed in field, one line
	JTextArea – more lines
	JEditorPane – more possibilities
	Timer – when a certain amount of time has passed
	JMenuItem – when a menu item is selected
	JLabel - can contain image
		More actionPerformed on ActionListeners
		
		//send ActionEvent on pressing Enter
		
		//void append(String t)
		Both: String getText() and void setText(String t)

Timer	Javax.swing.Timer Timer t = new Timer(1000, listener); t.start(); t.stop();	The Timer object sends actionPerformed events to its ActionListeners with regular intervals, thus: it only works when Swing Components are active.
JSlider	javax.swing JSlider(int orientation, int min, int max, int initValue) int getValue() addChangeListener(ChangeListener changeListener) stateChanged(ChangeEvent e)	A JSlider paints a Slider with a minimum and maximum value and a start value. This way the value of a variable can be changed by the user. //ChangeListener

2019 Q1 Programming Summary

Interface Collection <E>	boolean add(E e) – adds element to collection; true when collection changed void remove(E e) – removes element (not always implemented) boolean contains(E e) – true if element in collection boolean isEmpty(E e) – true if collection contains no elements int size() – returns number of elements in collection V put(K key, V value) – add entry (key, value) to the map or change entry if key already is in the map. return null resp. the old value V get(K key) – return the value corresponding the key, null if key not present Set<K> keySet() – returns the set of the keys in the collection Collection values() – returns collection of values; may contain duplicates int size(), V remove(K key)	<E> is the type parameter. You can do many things with these interfaces: some examples are shown on the left. Set<E> is a mathematical set which contains no duplicates. Implementation: HashSet<E>. List<E> is an ordered (not necessarily sorted) list. Implementation: ArrayList<E> can grow and shrink. Mathematical functions K → V (pairs) are alike: HashMap<K, V> (efficient but not thread-safe) and Hashtable<K,V> (thread-safe but less efficient) On the left you can also see some other functions which are of importance to mathematical functions from K to V.

Handling exception locally	try {
	<normal code>
	} catch(InputMismatchException e) {
	<code for when it goes wrong>
	}
	
	//to check for more exceptions
	try {
	...
	} catch (FileNotFoundException e1) {
	// if it is a FNFExc
	} catch (IOException e2) {
	// if it is a IOExc, but not a FNFExc
	} catch (Exception e3) {
	// if it is not a FNFExc
	// but any other exception
	}
		You use a try-block and a catch clause to handle an error instead of aborting it.
		
		Exceptions are actually objects with their classes ordered in hierarchy. Therefore it has a certain order for the catch clauses.
		
		Special class of exceptions need not to be declared. It can occur in many places, often not tied to a specific method, often due to (avoidable) programming errors, declaring all these would make program unwieldy
		Class is RuntimeException Examples:
		- IndexOutOfBoundsException: when using wrong index into array
		- ArithmeticException: division by zero etc., not overflow
		- NullPointerException: when method (or instance var) is accessed via null value

2019 Q1 Programming Summary

Files	File file = new File("funnyfile.txt");
	Scanner scanner = new Scanner(file);
	//creates a file object connected to funnyfile.txt
	//read all lines into array a:
	int i = 0;
	while (scanner.hasNext()) {
	a[i] = scanner.nextLine();
	i++;
	}
		Files store data more permanently. An object of class File records information about a file on disk.
		You can also connect a file to a path name:
		new File("C:\\My Documents\\funny.doc")

Recursion vs. iteration	//Iteration:
	int fac_iterative(int n) {
	int f = 1;
	while (n>0) {
	f = n * f;
	n = n-1;
	}
	return f;
	}
	//same but as recursive:
	int fac(int n) {
	if (n==0) {
	return 1;
	} else {
	return n * fac(n-1);
	} }
		Iteration is the opposite of recursion.
		In recursion, you divide the problem into simple base case(s) or problems of a smaller size. There is often reduction to several instances of the same problem.

2019 Q1 Programming Summary

Examples

Topic	The example	What it means

Print	// this prints out “Hi there”
	public class Greeter {
	void greet() {
	System.out.println("Hi there");
	}
	public static void main(String[] args){
	(new Greeter()).greet();
	}
	}
		comment (ignored)
		class declaration
		method declaration
		print statement with new line
		closing brace
		declaration of starter
		starter code
		
		class closing brace

Variables	class Variables {
	double amount;
	double interest;
	void showInterest() {
	amount = 1200.00;
	System.out.println("amount: " + amount);
	interest = 3.0;
	amount = amount * (100+interest)/100;
	System.out.println(“after a year: " + amount);
	amount = amount * (100+interest)/100;
	System.out.println("after 2 years: " + amount);
	
	public static void main(String[] args){
	(new Variables()).showInterest();
	}}
		
		Declaration
		Declaration (in a percentage!)
		
		Assignment
		Prints: amount: 1200.0
		Assignment
		
		Prints: after a year: 1236.0
		
		Prints: after 2 years: 1273.8
		
		Creates object and
		starts method showInterest

Scanner	import java.util.Scanner;
	class InputExample {
	Scanner scanner;
	String name;
	public void readNameAndAge() {
	scanner = new Scanner(System.in);
	System.out.print("Name? ");
	name = scanner.next();
	System.out.print("Age? ");
	age = scanner.nextInt();
	System.out.println("Hello "+name+
	", you are "+age+" springs young");
	}
	public static void main(String[] args) {
	new InputExample().readNameAndAge();
	} }
		Necessary to use Scanner
		
		Create scanner
		
		Connect scanner to console
		
		Read next word from input
		
		Read an integer from input

While-construct	int x = 0;
	while (x<3) {
	System.out.println(x);
	x = x+1;
	}
	System.out.println("ready!");
		Let x = 0
		If x is still smaller than 3, continue.
		Print out x
		And add 1 to x
		
		If x is bigger than 3, print “ready!”
		gives: 0 1 2 ready!

2019 Q1 Programming Summary

For-loop	for (int i = 0; i<100; i=i+1) {
	System.out.println(
	"I should not forget the around loop bodies."
	);
	}
		If i start with the value 0, i should be smaller than 100 and we add 1 to i every loop, then we will print the sentence 100 times.

Do-while loop	double balance;
	String choice;
	balance = 0;
	do {
	System.out.println("Type amount for deposit.");
	deposit = sc.nextDouble();
	balance = balance + deposit;
	System.out.println("New: " + balance + ".");
	System.out.println(
	"Do you want to make another deposit? Type y/n");
	choice = sc.next());
	} while (choice.equals("y"));
		You do the following statements.
		
		We check whether the condition is true: if so, we repeat the statements.

Array with while- or for-loop	int[] students;
	double sum;
	int i;
	students = new int[3];
	students[0] = 7;
	students[1] = 5;
	students[2] = 6;
	With while-loop:
	sum = 0;
	i = 0;
	while (i<3) {
	sum = sum + students[i];
	i = i+1;
	}
	With for-loop:
	sum = 0.0;
	for (int i=0; i<3; i++) {
	sum = sum + students[i];
	}
	double average = sum/3;
		Collects the sum
		Is the index variable
		This is the array
		
		We calculate the sum
		
		While i is smaller than 3,
		We add another number to the sum, dependent on the i
		
		When using the variable n (assigned by a scanner) instead of the number 3, the program becomes more flexible: now the program works for any number of students. You can also use students.length for n.

For-each loop	int[] grades = new int[3];
	grades[0] = 7;
	grades[1] = 5;
	grades[2] = 6;
	double sum = 0.0;
	Use for-each loop:
	for (int grade : grades) {
	sum = sum + grade; }
	Instead of for loop:
	for (int i=0; i<grades.length; i++) {
	sum = sum + grades[i]; }
		This shows that using a for-each loop is shorter than a for loop.

2019 Q1 Programming Summary

Array to store frequency table	Scanner scanner = new Scanner(System.in);
	int[] frequencies; //frequency table of grades
	int grade; //grade just read
	
	void calculateFrequencies() {
	  frequencies = new int[11];
	//fill frequency table until there is no grade
	  int grade = 0;
	  while(0 <= grade && grade <= 10) {
	  grade = scanner.nextInt();
	  frequencies[grade] += 1;
	} }
	
	void printFrequencies() {
	  //print frequency table
	  for (int g=0; g<frequencies.length; g++) {
	  System.out.println("grade "+g+" occurs "
	  +frequencies[g]+" time(s)");
	} }
		This array is used to make a histogram of the grades.
		
		Here we fill the array with the gives grades.
		
		And here we print out the array.

Example HashMap (Hashtable)	Map<String, Integer> table = new HashMap<>();
	Scanner sc;
	void loadTable(String filename) {
	  File text = new File(filename);
	  try {
	  sc = new Scanner(text);
	  sc.useDelimiter(\\W+);
	  while (sc.hasNext()) {
	  String word = sc.next().toLowerCase();
	  if (table.containsKey(word)) {
	  table.put(word, table.get(word) + 1);
	  } else {
	  table.put(word, 1);
	  }
	  }
	  sc.close();
	  } catch(FileNotFoundException e) {
	  System.err.println("No such file\n"+ e);
	  }
	}
	void showBasicTable() {
	  Set<String> keys = table.keySet();
	  for (String word : keys) {
	  System.out.println(word+" "+table.get(word)
	  }
	  System.out.println();
	}
		//or Hashtable
		
		//to get words without interpunction

2019 Q1 Programming Summary

Handling exception locally	void readMatch() {
	do {
	try {
	...
	goalsFor = sc.nextInt();
	ok = true;
	} catch(InputMismatchException e) {
	System.out.println(
	“Wrong input, enter a number”);
	ok = false;
	}
	} while (!ok);
	... continue with rest of input
	}
		/*This part is not executed in case of
		*exception
		*/
		
		/*This part is only executed in case of
		*exception
		*/

Passing an exception to caller	void readMatch() throws
	InputMismatchException {
	...
	goalsFor = sc.nextInt();
	goalsAgainst = sc.nextInt();
	matches += 1;
	...
	}
	void readAll() {
	while (...) {
	try {
	readMatch();
	} catch(InputMismatchException e) {
	System.out.println(e + “Match input ignored”);
	}
	}
	}
		This code throws the clause: the method is aborted but the exception is passed to the caller.
		//this part is not executed in case of exception
		
		//this part is only executed in case of exception

2019 Q1 Programming Summary

Files	import java.util.*;
	import java.io.*;
	public class FileCopy {
	Scanner scanner;
	File source;
	File target;
	PrintWriter pw;
	String filenameSource = "rhubarb.txt";
	String filenameTarget = "rhubarb_copy.txt";
	void copy() {
	try {
	String line = null;
	source = new File(filenameSource) ;
	scanner = new Scanner(source);
	target = new File(filenameTarget);
	pw = new PrintWriter(target);
	while (scanner.hasNext()) {
	line = scanner.nextLine();
	pw.println(line);
	}
	pw.close();
	} catch(FileNotFoundException e) {
	System.out.println("Could not open file due to");
	System.out.println(e);
	}
	}
	public static void main(String[] args) {
	new FileCopy().copy();
	}
	}
		// copies a text file line by line

Multithreading	public static void main(String[] a) {
	(new MultipleThreadShowIL()).loop();
	}
	MultipleThreadShowIL() {
	SwingUtilities.invokeLater(() -> {
	JFrame f = new JFrame("SwingThreadShow");
	f.setSize(300, 300);
	f.add(MultipleThreadShowIL.this);
	f.addMouseListener(MultipleThreadShowIL.this
	);
	f.setVisible(true);
	});
	}
		Use invokeLater. This way you don't get problems with multithreading.
		This method schedules code for execution in the Swing thread
		Pass this code as body of run in Interface Runnable

2019 Q1 Programming Summary

SwingWorker	@Override
	public void mousePressed(MouseEvent me) {
	  new SwingWorkertje().execute();
	}
	...
	
	class SwingWorkertje extends
	SwingWorker<Integer, String> {
	  @Override
	  public Integer doInBackground() {
	  int prod = 1;
	  for (int i=1; i<10; i++) {
	  System.out.println("i: "+i);
	  prod *= i;
	  try{Thread.sleep(500);}
	  catch(InterruptedException e){}
	  }
	  return prod;
	  }
	  @Override
	  public void done() {
	  try {
	  n = get();
	  } catch(Exception e) {}
	  repaint();
	  }
	}
		What if we do have time consuming code that has to be triggered by a handler? Subclass SwingWorker and put your code there. A SwingWorker executes code “in the background” (in another thread). When code is finished done is called in the Swing thread
		
		//return type and type for intermediate messages
		//called when execute is called
		
		//called in Swing thread when doInBackground is finished

2019 Q1 Programming Summary

Most occurring errors:

- Initialization: It hasn't been initialized

2019 Q1 Programming Summary

Article:

1. "… Expected"

This error occurs when something is missing from the code. Often this is created by a missing semicolon or closing parenthesis.

```
private static double volume(String solidom, double alturam, double areaBasem, double
raiom) {
double vol;
   if (solidom.equalsIgnoreCase("esfera"){
      vol=(4.0/3)*Math.pi*Math.pow(raiom,3);
   }
   else {
      if (solidom.equalsIgnoreCase("cilindro") {
         vol=Math.pi*Math.pow(raiom,2)*alturam;
      }
      else {
         vol=(1.0/3)*Math.pi*Math.pow(raiom,2)*alturam;
      }
   }
   return vol;
}
```

Often this error message does not pinpoint the exact location of the issue. To find it:

- Make sure all opening parenthesis have a corresponding closing parenthesis.
- Look in the line previous to the Java code line indicated. This Java software error doesn't get noticed by the compiler until further in the code.
- Sometimes a character such as an opening parenthesis shouldn't be in the Java code in the first place. So the developer didn't place a closing parenthesis to balance the parentheses.

Check out an example of how a missed parenthesis can create an error (@StackOverflow).

2. "Unclosed String Literal"

The "unclosed string literal" error message is created when the string literal ends without quotation marks, and the message will appear on the same line as the error. (@DreamInCode) A literal is a source code of a value.

```
 public abstract class NFLPlayersReference {
   private static Runningback[] nflplayersreference;
   private static Quarterback[] players;
   private static WideReceiver[] nflplayers;
   public static void main(String args[]){
   Runningback r = new Runningback("Thomlinsion");
   Quarterback q = new Quarterback("Tom Brady");
   WideReceiver w = new WideReceiver("Steve Smith");
   NFLPlayersReference[] NFLPlayersReference;
      Run();// {
      NFLPlayersReference = new NFLPlayersReference [3];
      nflplayersreference[0] = r;
      players[1] = q;
      nflplayers[2] = w;
         for ( int i = 0; i < nflplayersreference.length; i++ ) {
```

```
            System.out.println("My  name is " + " nflplayersreference[i].getName());
            nflplayersreference[i].run();
            nflplayersreference[i].run();
            nflplayersreference[i].run();
            System.out.println("NFL  offensive threats have great running abilities!");
        }
    }
    private static void Run() {
        System.out.println("Not  yet implemented");
    }
}
```

Commonly, this happens when:

- The string literal does not end with quote marks. This is easy to correct by closing the string literal with the needed quote mark.
- The string literal extends beyond a line. Long string literals can be broken into multiple literals and concatenated with a plus sign ("+").
- Quote marks that are part of the string literal are not escaped with a backslash ("\").

Read a discussion of the unclosed string literal Java software error message. (@Quora)

3. "Illegal Start of an Expression"

There are numerous reasons why an "illegal start of an expression" error occurs. It ends up being one of the less-helpful error messages. Some developers say it's caused by bad code. Usually, expressions are created to produce a new value or assign a value to a variable. The compiler expects to find an expression and cannot find it because the syntax does not match expectations. (@StackOverflow) It is in these statements that the error can be found.

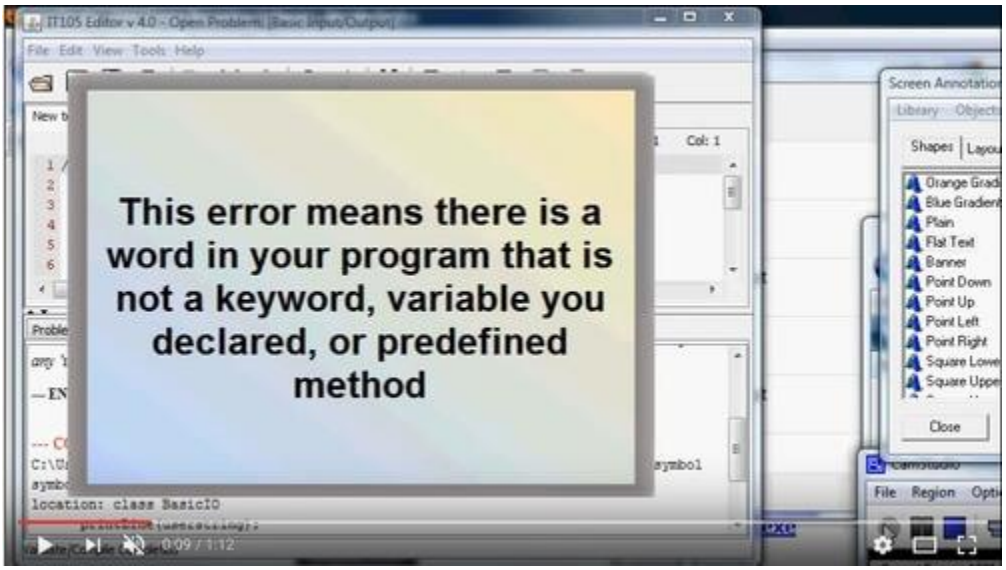
```
} // ADD IT HERE
      public void newShape(String shape) {
       switch (shape) {
          case "Line":
             Shape line = new Line(startX, startY, endX, endY);
          shapes.add(line);
          break;
             case "Oval":
          Shape oval = new Oval(startX, startY, endX, endY);
          shapes.add(oval);
          break;
          case "Rectangle":
          Shape rectangle = new Rectangle(startX, startY, endX, endY);
          shapes.add(rectangle);
          break;
          default:
          System.out.println("ERROR.  Check logic.");
       }
       }
    } // REMOVE IT FROM HERE
    }
```

Browse discussions of how to troubleshoot the "illegal start of an expression" error. (@StackOverflow)

2019 Q1 Programming Summary

4. "Cannot Find Symbol"

This is a very common issue because all identifiers in Java need to be declared before they are used. When the code is being compiled, the compiler does not understand what the identifier means.



There are many reasons you might receive the "cannot find symbol" message:

- The spelling of the identifier when declared may not be the same as when it is used in the code.
- The variable was never declared.
- The variable is not being used in the same scope it was declared.
- The class was not imported.

Read a thorough discussion of the "cannot find symbol" error and examples of code that create this issue. (@StackOverflow)

5. "Public Class XXX Should Be in File"

The "public class XXX should be in file" message occurs when the class XXX and the Java program filename do not match. The code will only be compiled when the class and Java file are the same. (@coderanch):

```
package javaapplication3;
 public class Robot {
      int xlocation;
      int ylocation;
      String name;
      static int ccount = 0;
      public Robot(int xxlocation, int yylocation, String nname) {
         xlocation = xxlocation;
         ylocation = yylocation;
         name = nname;
         ccount++;
      }
 }
 public class JavaApplication1 {
   public static void main(String[] args) {
      robot firstRobot = new Robot(34,51,"yossi");
      System.out.println("numebr of robots is now " + Robot.ccount);
   }
```

```
}
```

To fix this issue:

- Name the class and file the same.
- Make sure the case of both names is consistent.

See an example of the "Public class XXX should be in file" error. (@StackOverflow)

6. "Incompatible Types"

"Incompatible types" is an error in logic that occurs when an assignment statement tries to pair a variable with an expression of types. It often comes when the code tries to place a text string into an integer — or vice versa. This is not a Java syntax error. (@StackOverflow)

```
test.java:78: error: incompatible types
return stringBuilder.toString();
                    ^
required: int
found:    String
1 error
```

There really isn't an easy fix when the compiler gives an "incompatible types" message:

- There are functions that can convert types.
- The developer may need change what the code is expected to do.

Check out an example of how trying to assign a string to an integer created the "incompatible types." (@StackOverflow)

7. "Invalid Method Declaration; Return Type Required"

This Java software error message means the return type of a method was not explicitly stated in the method signature.

```
public class Circle
{
   private double radius;
   public CircleR(double r)
   {
      radius = r;
   }
   public diameter()
   {
     double d = radius * 2;
     return d;
   }
}
```

There are a few ways to trigger the "invalid method declaration; return type required" error:

- Forgetting to state the type
- If the method does not return a value then "void" needs to be stated as the type in the method signature.
- Constructor names do not need to state type. But if there is an error in the constructor name, then the compiler will treat the constructor as a method without a stated type.

Follow an example of how constructor naming triggered the "invalid method declaration; return type required" issue. (@StackOverflow)

8. "Method <X> in Class <Y> Cannot Be Applied to Given Types"

This Java software error message is one of the more helpful error messages. It explains how the method signature is calling the wrong parameters.

RandomNumbers.java:9: error: method generateNumbers in class RandomNumbers cannot be applied to given types;
generateNumbers();
required: int[]
found:generateNumbers();
reason: actual and formal argument lists differ in length

The method called is expecting certain arguments defined in the method's declaration. Check the method declaration and call carefully to make sure they are compatible.

This discussion illustrates how a Java software error message identifies the incompatibility created by arguments in the method declaration and method call. (@StackOverflow)

9. "Missing Return Statement"

The "missing return statement" message occurs when a method does not have a return statement. Each method that returns a value (a non-void type) must have a statement that literally returns that value so it can be called outside the method.

```
public String[] OpenFile() throws IOException {
   Map<String, Double> map = new HashMap();
   FileReader fr = new FileReader("money.txt");
   BufferedReader br = new BufferedReader(fr);
   try{
      while (br.ready()){
         String str = br.readLine();
         String[] list = str.split(" ");
         System.out.println(list);
      }
   }  catch (IOException e){
      System.err.println("Error - IOException!");
   }
}
```

There are a couple reasons why a compiler throws the "missing return statement" message:

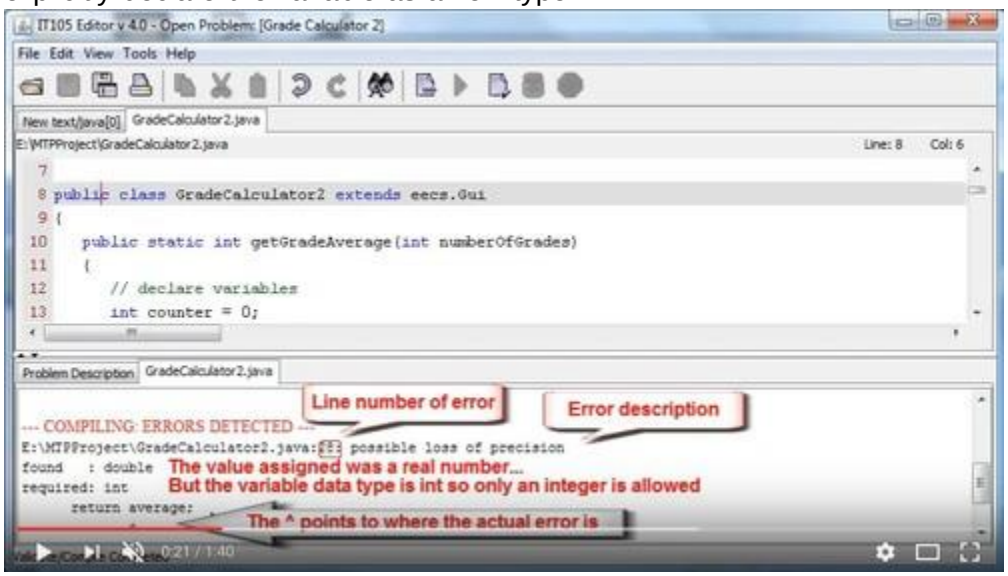
- A return statement was simply omitted by mistake.
- The method did not return any value but type void was not declared in the method signature.

Check out an example of how to fix the "missing return statement" Java software error. (@StackOverflow)

2019 Q1 Programming Summary

10. "Possible Loss of Precision"

"Possible loss of precision" occurs when more information is assigned to a variable than it can hold. If this happens, pieces will be thrown out. If this is fine, then the code needs to explicitly declare the variable as a new type.



A "possible loss of precision" error commonly occurs when:

- Trying to assign a real number to a variable with an integer data type.
- Trying to assign a double to a variable with an integer data type.

This explanation of Primitive Data Types in Java shows how the data is characterized. (@Oracle)

11. "Reached End of File While Parsing"

This error message usually occurs in Java when the program is missing the closing curly brace ("}"). Sometimes it can be quickly fixed by placing it at the end of the code.

```
public class mod_MyMod extends BaseMod
public String Version()
{
    return "1.2_02";
}
public void AddRecipes(CraftingManager recipes)
{
  recipes.addRecipe(new ItemStack(Item.diamond), new Object[] {
    "#", Character.valueOf('#'), Block.dirt
  });
}
```

The above code results in the following error:
java:11: reached end of file while parsing }

Coding utilities and proper code indenting can make it easier to find these unbalanced braces.

This example shows how missing braces can create the "reached end of file while parsing" error message. (@StackOverflow)

12. "Unreachable Statement"

"Unreachable statement" occurs when a statement is written in a place that prevents it from being executed. Usually, this is after a break or return statement.

```
for(;;){
  break;
  ... // unreachable statement
}
int i=1;
if(i==1)
 ...
else
 ... // dead code
```

Often simply moving the return statement will fix the error. Read the discussion of how to fix unreachable statement Java software error. (@StackOverflow)

13. "Variable <X> Might Not Have Been Initialized"

This occurs when a local variable declared within a method has not been initialized. It can occur when a variable without an initial value is part of an if statement.

```
int x;
if (condition) {
   x = 5;
}
System.out.println(x);  // x may not have been initialized
```

Read this discussion of how to avoid triggering the "variable <X> might not have been initialized" error. (@reddit)

14. "Operator ... Cannot be Applied to <X>"

This issue occurs when operators are used for types not in their definition.

```
operator < cannot be applied to java.lang.Object,java.lang.Object
```

This often happens when the Java code tries to use a type string in a calculation. To fix it, the string needs to be converted to an integer or float.

Read this example of how non-numeric types were causing a Java software error warning that an operator cannot be applied to a type. (@StackOverflow)

15. "Inconvertible Types"

The "inconvertible types" error occurs when the Java code tries to perform an illegal conversion.

```
TypeInvocationConversionTest.java:12:  inconvertible types
found   : java.util.ArrayList<java.lang.Class<?  extends
TypeInvocationConversionTest.Interface1>>
required: java.util.ArrayList<java.lang.Class<?>>
   lessRestrictiveClassList  = (ArrayList<Class<?>>) classList;
                                        ^
```

For example, booleans cannot be converted to an integer.

Read this discussion about finding ways to convert inconvertible types in Java software. (@StackOverflow)

16. "Missing Return Value"

You'll get the "missing return value" message when the return statement includes an incorrect type. For example, the following code:

```
public class SavingsAcc2 {
   private double balance;
   private double interest;
   public SavingsAcc2() {
      balance = 0.0;
      interest = 6.17;
   }
   public SavingsAcc2(double initBalance, double interested) {
      balance = initBalance;
      interest = interested;
   }
   public SavingsAcc2 deposit(double amount) {
      balance = balance + amount;
      return;
   }
   public SavingsAcc2 withdraw(double amount) {
      balance = balance - amount;
      return;
   }
   public SavingsAcc2 addInterest(double interest) {
      balance = balance * (interest / 100) + balance;
      return;
   }
   public double getBalance() {
      return balance;
   }
}
```

Returns the following error:

```
SavingsAcc2.java:29: missing return value
return;
^
SavingsAcc2.java:35: missing return value
return;
^
SavingsAcc2.java:41: missing return value
return;
^
3 errors
```

Usually, there is a return statement that doesn't return anything.
Read this discussion about how to avoid the "missing return value" Java software error message. (@coderanch)

17. "Cannot Return a Value From Method Whose Result Type Is Void"

This Java error occurs when a void method tries to return any value, such as in the following example:

```
public static void move()
{
   System.out.println("What  do you want to do?");
   Scanner scan = new Scanner(System.in);
   int userMove = scan.nextInt();
   return userMove;
}
public static void usersMove(String playerName, int gesture)
{
   int userMove = move();
   if (userMove == -1)
   {
      break;
   }
```

Often this is fixed by changing to method signature to match the type in the return statement. In this case, instances of void can be changed to int:

```
public static int move()
{
   System.out.println("What  do you want to do?");
   Scanner scan = new Scanner(System.in);
   int userMove = scan.nextInt();
   return userMove;
}
```

Read this discussion about how to fix the "cannot return a value from method whose result type is void" error. (@StackOverflow)

18. "Non-Static Variable ... Cannot Be Referenced From a Static Context"

This error occurs when the compiler tries to access non-static variables from a static method (@javinpaul):

```
public class StaticTest {
   private int count=0;
   public static void main(String args[]) throws IOException {
      count++; //compiler error: non-static variable count cannot be referenced from a static
context
   }
}
```

To fix the "non-static variable ... cannot be referenced from a static context" error, two things can be done:

- The variable can be declared static in the signature.
- The code can create an instance of a non-static object in the static method.

Read this tutorial that explains what is the difference between static and non-static variables. (@sitesbay)

19. "Non-Static Method ... Cannot Be Referenced From a Static Context"

This issue occurs when the Java code tries to call a non-static method in a non-static class. For example, the following code:

```
class Sample
{
  private int age;
  public void setAge(int a)
  {
    age=a;
  }
  public int getAge()
  {
    return age;
  }
  public static void main(String args[])
  {
     System.out.println("Age is:"+ getAge());
  }
}
```

Would return this error:
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Cannot make a static reference to the non-static method getAge() from the type Sample

To call a non-static method from a static method is to declare an instance of the class calling the non-static method.
Read this explanation of what is the difference between non-static methods and static methods.

20. "(array) <X> Not Initialized"

You'll get the "(array) <X> not initialized" message when an array has been declared but not initialized. Arrays are fixed in length so each array needs to be initialized with the desired length.
The following code is acceptable:

```
AClass[] array = {object1, object2}
```

As is:

```
AClass[] array = new AClass[2];
...
array[0] = object1;
array[1] = object2;
```

But not:

```
AClass[] array;
...
array = {object1, object2};
```

Read this discussion of how to initialize arrays in Java software. (@StackOverflow)

21. "ArrayIndexOutOfBoundsException"

This is a runtime error message that occurs when the code attempts to access an array index that is not within the values. The following code would trigger this exception:

```
String[] name = {
   "tom",
   "dick",
   "harry"
};
for (int i = 0; i <= name.length; i++) {
   System.out.print(name[i] + '\n');
}
```

Here's another example (@DukeU):

```
int[] list = new int[5];
list[5] = 33; // illegal index, maximum index is 4
```

Array indexes start at zero and end at one less than the length of the array. Often it is fixed by using "<" instead of "<=" when defining the limits of the array index.
Check out this example of how an index triggered the "ArrayIndexOutOfBoundsException" Java software error message. (@StackOverflow)

22. "StringIndexOutOfBoundsException"

This is an issue that occurs when the code attempts to access part of the string that is not within the bounds of the string. Usually, this happens when the code tries to create a substring of a string that is not as long as the parameters are set at. Here's an example (@javacodegeeks):

```
public class StringCharAtExample {
   public static void main(String[] args) {
      String str = "Java Code Geeks!";
      System.out.println("Length: " + str.length());
      //The following statement throws an exception, because
      //the request index is invalid.
      char ch = str.charAt(50);
   }
}
```

Like array indexes, string indexes start at zero. When indexing a string, the last character is at one less than the length of the string. The "StringIndexOutOfBoundsException" Java software error message usually means the index is trying to access characters that aren't there.
Here's an example that illustrates how the "StringIndexOutOfBoundsException" can occur and be fixed. (@StackOverflow)

23. "NullPointerException"

A "NullPointerException" will occur when the program tries to use an object reference that does not have a value assigned to it (@geeksforgeeks).

```
// A Java program to demonstrate that invoking a method
// on null causes NullPointerException
import java.io.*;
class GFG
{
   public static void main (String[] args)
   {
      // Initializing String variable with null value
      String ptr = null;
      // Checking if ptr.equals null or works fine.
      try
      {
         // This line of code throws NullPointerException
         // because ptr is null
         if (ptr.equals("gfg"))
            System.out.print("Same");
         else
            System.out.print("Not Same");
      }
      catch(NullPointerException e)
      {
         System.out.print("NullPointerException Caught");
      }
   }
}
```

The Java program raises an exception often when:

- A statement references an object with a null value.
- Trying to access a class that is defined but isn't assigned a reference.

Here's discussion of when developers may encounter the "NullPointerException" and how to handle it. (@StackOverflow)

24. "NoClassDefFoundError"

The "NoClassDefFoundError" will occur when the interpreter cannot find the file containing a class with the main method. Here's an example from DZone (@DZone):

If you compile this program:

```
class A
{
  // some code
}
public class B
{
   public static void main(String[] args)
   {
      A a = new A();
} }
```

Two .class files are generated: A.class and B.class. Removing the A.class file and running the B.class file, you'll get the NoClassDefFoundError:

```
Exception in thread "main" java.lang.NoClassDefFoundError: A
at MainClass.main(MainClass.java:10)
Caused by: java.lang.ClassNotFoundException: A
at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
```

This can happen if:

- The file is not in the right directory.
- The name of the class must be the same as the name of the file (without the file extension). The names are case sensitive.

Read this discussion of why "NoClassDefFoundError" occurs when running Java software. (@StackOverflow)

25. "NoSuchMethodFoundError"

This error message will occur when the Java software tries to call a method of a class and the method no longer has a definition (@myUND):

```
Error: Could not find or load main class wiki.java
```

Often the "NoSuchMethodFoundError" Java software error occurs when there is a typo in the declaration.

Read this tutorial to learn how to avoid the error message NoSuchMethodFoundError." (@javacodegeeks)

26. "NoSuchProviderException"

"NoSuchProviderException" occurs when a security provider is requested that is not available (@alvinalexander): javax.mail.NoSuchProviderException

When trying to find why "NoSuchProviderException" occurs, check:

- The JRE configuration.
- The Java home is set in the configuration.
- Which Java environment is used.
- The security provider entry.

Read this discussion of what causes "NoSuchProviderException" when Java software is run. (@StackOverflow)

27. AccessControlException

AccessControlException indicates that requested access to system resources such as a file system or network is denied, as in this example from JBossDeveloper (@jbossdeveloper):
ERROR Could not register mbeans java.security.
AccessControlException: WFSM000001: Permission check failed (permission "("javax.management.MBeanPermission" "org.apache.logging.log4j.core.jmx.LoggerContextAdmin#-[org.apache.logging.log4j2:type=51634f]" "registerMBean")" in code source "(vfs:/C:/wildfly-10.0.0.Final/standalone/deployments/mySampleSecurityApp.war/WEB-INF/lib/log4j-core-2.5.
jar)" of "null")

Read this discussion of a workaround used to get past an "AccessControlException" error. (@github)

28. "ArrayStoreException"

An "ArrayStoreException" occurs when the rules of casting elements in Java arrays are broken. Arrays are very careful about what can go into them. (@Roedyg) For instance, this example from JavaScan.com illustrates that this program (@java_scan):

```
/* ..... START ..... */
public class JavaArrayStoreException {
    public static void main(String...args) {
        Object[] val = new Integer[4];
        val[0] = 5.8;
    }
}
/* ..... END ..... */
```

Results in the following output:
Exception in thread "main" java.lang.ArrayStoreException: java.lang.Double
at ExceptionHandling.JavaArrayStoreException.main(JavaArrayStoreException.java:7)

When an array is initialized, the sorts of objects allowed into the array need to be declared. Then each array element needs be of the same type of object.
Read this discussion of how to solve for the "ArrayStoreException." (@StackOverflow)

2019 Q1 Programming Summary

29. "Bad Magic Number"

This Java software error message means something may be wrong with the class definition files on the network. Here's an example from The Server Side (@TSS_dotcom):

```
Java(TM) Plug-in: Version 1.3.1_01
Using JRE version 1.3.1_01 Java HotSpot(TM) Client VM
User home directory = C:\Documents and Settings\Ankur
Proxy Configuration: Manual Configuration
Proxy: 192.168.11.6:80
java.lang.ClassFormatError: SalesCalculatorAppletBeanInfo (Bad magic number)
at java.lang.ClassLoader.defineClass0(Native Method)
at java.lang.ClassLoader.defineClass(Unknown Source)
at java.security.SecureClassLoader.defineClass(Unknown Source)
at sun.applet.AppletClassLoader.findClass(Unknown Source)
at sun.plugin.security.PluginClassLoader.access$201(Unknown Source)
at sun.plugin.security.PluginClassLoader$1.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at sun.plugin.security.PluginClassLoader.findClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
at sun.applet.AppletClassLoader.loadClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
at java.beans.Introspector.instantiate(Unknown Source)
at java.beans.Introspector.findInformant(Unknown Source)
at java.beans.Introspector.(Unknown Source)
at java.beans.Introspector.getBeanInfo(Unknown Source)
at sun.beans.ole.OleBeanInfo.(Unknown Source)
at sun.beans.ole.StubInformation.getStub(Unknown Source)
at sun.plugin.ocx.TypeLibManager$1.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at sun.plugin.ocx.TypeLibManager.getTypeLib(Unknown Source)
at sun.plugin.ocx.TypeLibManager.getTypeLib(Unknown Source)
at sun.plugin.ocx.ActiveXAppletViewer.statusNotification(Native Method)
at sun.plugin.ocx.ActiveXAppletViewer.notifyStatus(Unknown Source)
at sun.plugin.ocx.ActiveXAppletViewer.showAppletStatus(Unknown Source)
at sun.applet.AppletPanel.run(Unknown Source)
at java.lang.Thread.run(Unknown Source)
```

The "bad magic number" error message could happen when:

- The first four bytes of a class file is not the hexadecimal number CAFEBABE.
- The class file was uploaded as in ASCII mode not binary mode.
- The Java program is run before it is compiled.

Read this discussion of how to find the reason for a "bad magic number." (@coderanch)

30. "Broken Pipe"

This error message refers to the data stream from a file or network socket has stopped working or is closed from the other end (@ExpertsExchange).

```
Exception in thread "main" java.net.SocketException: Broken pipe
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:92)
    at java.net.SocketOutputStream.write(SocketOutputStream.java:115)
    at java.io.DataOutputStream.write
```

The causes of a broken pipe often include:

- Running out of disk scratch space.
- RAM may be clogged.
- The datastream may be corrupt.
- The process reading the pipe might have been closed.

Read this discussion of what is the Java error "broken pipe." (@StackOverflow)

31. "Could Not Create Java Virtual Machine"

This Java error message usually occurs when the code tries to invoke Java with the wrong arguments (@ghacksnews):

```
Error: Could not create the Java Virtual Machine
Error: A fatal exception has occurred. Program will exit.
```

It often is caused by a mistake in the declaration in the code or allocating the proper amount of memory to it.

Read this discussion of how to fix the Java software error "Could not create Java Virtual Machine." (@StackOverflow)

32. "class file contains wrong class"

The "class file contains wrong class" issue occurs when the Java code tries to find the class file in the wrong directory, resulting in an error message similar to the following:

```
MyTest.java:10: cannot access MyStruct
bad class file: D:\Java\test\MyStruct.java
file does not contain class MyStruct
Please remove or make sure it appears in the correct subdirectory of the classpath.
MyStruct ms = new MyStruct();
^
```

To fix this error, these tips could help:

- Make sure the name of the source file and the name of the class match — including case.
- Check if the package statement is correct or missing.
- Make sure the source file is in the right directory.

Read this discussion of how to fix a "class file contains wrong class" error. (@StackOverflow)

33. "ClassCastException"

The "ClassCastException" message indicates the Java code is trying to cast an object to the wrong class. In this example from Java Concept of the Day, running the following program:

```
package com;
class A
{
   int i = 10;
}
class B extends A
{
   int j = 20;
}
class C extends B
{
   int k = 30;
}
public class ClassCastExceptionDemo
{
   public static void main(String[] args)
   {
      A a = new B();   //B type is auto up casted to A type
      B b = (B) a;     //A type is explicitly down casted to B type.
      C c = (C) b;    //Here, you will get class cast exception
      System.out.println(c.k);
   }
}
```

Results in this error:

```
Exception in thread "main" java.lang.ClassCastException: com.B cannot be cast to com.C
at com.ClassCastExceptionDemo.main(ClassCastExceptionDemo.java:23)
```

The Java code will create a hierarchy of classes and subclasses. To avoid the "ClassCastException" error, make sure the new type belongs to the right class or one of its parent classes. If Generics are used, these errors can be caught when the code is compiled. Read this tutorial on how to fix "ClassCastException" Java software errors. (@java_concept)

2019 Q1 Programming Summary

34. "ClassFormatError"

The "ClassFormatError" message indicates a linkage error and occurs when a class file cannot be read or interpreted as a class file.

```
Caused by: java.lang.ClassFormatError: Absent Code attribute in method that is
      not native or abstract in class file javax/persistence/GenerationType
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClassCond(Unknown Source)
at java.lang.ClassLoader.defineClass(Unknown Source)
at java.security.SecureClassLoader.defineClass(Unknown Source)
at java.net.URLClassLoader.defineClass(Unknown Source)
at java.net.URLClassLoader.access$000(Unknown Source)
at java.net.URLClassLoader$1.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
```

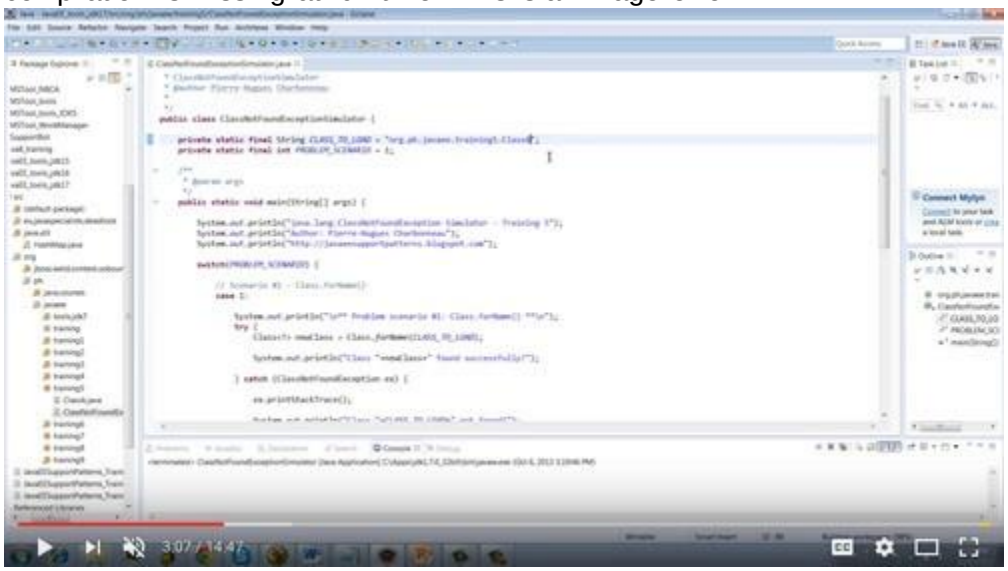
There are several reasons why a "ClassFormatError" can occur:

- The class file was uploaded as in ASCII mode not binary mode.
- The web server must send class files as binary not ASCII.
- There could be a classpath error that prevents the code from finding the class file.
- If the class is loaded twice, the second time will cause the exception to be thrown.
- An old version of Java runtime is being used.

Read this discussion about what causes the "ClassFormatError" in Java. (@StackOverflow)

35. "ClassNotFoundException"

"ClassNotFoundException" only occurs at run time — meaning a class that was there during compilation is missing at run time. This is a linkage error.



Much like the "NoClassDefFoundError," this issue can occur if:

- The file is not in the right directory.
- The name of the class must be the same as the name of the file (without the file extension). The names are case sensitive.

Read this discussion of what causes "ClassNotFoundException" the for more cases. (@StackOverflow).

36. "ExceptionInInitializerError"

This Java issue will occur when something goes wrong with a static initialization (@GitHub). When the Java code later uses the class, the "NoClassDefFoundError" error will occur.

```
java.lang.ExceptionInInitializerError
  at org.eclipse.mat.hprof.HprofIndexBuilder.fill(HprofIndexBuilder.java:54)
  at org.eclipse.mat.parser.internal.SnapshotFactory.parse(SnapshotFactory.java:193)
  at
org.eclipse.mat.parser.internal.SnapshotFactory.openSnapshot(SnapshotFactory.java:106)
  at com.squareup.leakcanary.HeapAnalyzer.openSnapshot(HeapAnalyzer.java:134)
  at com.squareup.leakcanary.HeapAnalyzer.checkForLeak(HeapAnalyzer.java:87)
  at
com.squareup.leakcanary.internal.HeapAnalyzerService.onHandleIntent(HeapAnalyzerServi
ce.java:56)
  at android.app.IntentService$ServiceHandler.handleMessage(IntentService.java:65)
  at android.os.Handler.dispatchMessage(Handler.java:102)
  at android.os.Looper.loop(Looper.java:145)
  at android.os.HandlerThread.run(HandlerThread.java:61)
Caused by: java.lang.NullPointerException:  in == null
  at java.util.Properties.load(Properties.java:246)
  at org.eclipse.mat.util.MessageUtil.(MessageUtil.java:28)
  at org.eclipse.mat.util.MessageUtil.(MessageUtil.java:13)
  ... 10 more
```

There needs to be more information to fix the error. Using *getCause()* in the code can return the exception that caused the error to be returned.
Read this discussion about how to track down the cause of the ExceptionInInitializerError. (@StackOverflow)

37. "IllegalBlockSizeException"

An "IllegalBlockSizeException" will occur during decryption when the length message is not a multiple of 8 bytes. Here's an example from ProgramCreek.com (@ProgramCreek):

```
@Override
protected byte[] engineWrap(Key key) throws IllegalBlockSizeException,
InvalidKeyException {
   try {
      byte[] encoded = key.getEncoded();
      return engineDoFinal(encoded, 0, encoded.length);
   } catch (BadPaddingException e) {
      IllegalBlockSizeException newE = new IllegalBlockSizeException();
      newE.initCause(e);
      throw newE;
   }
}
```

The "IllegalBlockSizeException" could be caused by:

- Different encryption and decryption algorithm options used.
- The message to be decrypted could be truncated or garbled in transmission.

Read this discussion about how to prevent the IllegalBlockSizeException Java software error message. (@StackOverflow)

38. "BadPaddingException"

A "BadPaddingException" will occur during decryption when padding was used to create a message than can be measured by a multiple of 8 bytes. Here's an example from Stack Overflow (@StackOverflow):

```
javax.crypto.BadPaddingException: Given final block not properly padded
at com.sun.crypto.provider.SunJCE_f.b(DashoA13*..)
at com.sun.crypto.provider.SunJCE_f.b(DashoA13*..)
at com.sun.crypto.provider.AESCipher.engineDoFinal(DashoA13*..)
at javax.crypto.Cipher.doFinal(DashoA13*..)
```

Encrypted data is binary so don't try to store it in a string or the data was not padded properly during encryption.

Read this discussion about how to prevent the BadPaddingException. (@StackOverflow)

39. "IncompatibleClassChangeError"

An "IncompatibleClassChangeError" is a form of LinkageError that can occur when a base class changes after the compilation of a child class. This example is from How to Do in Java (@HowToDoInJava):

```
Exception in thread "main" java.lang.IncompatibleClassChangeError: Implementing class
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(Unknown Source)
at java.security.SecureClassLoader.defineClass(Unknown Source)
at java.net.URLClassLoader.defineClass(Unknown Source)
at java.net.URLClassLoader.access$000(Unknown Source)
at java.net.URLClassLoader$1.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
at java.lang.ClassLoader.loadClassInternal(Unknown Source)
at net.sf.cglib.core.DebuggingClassWriter.toByteArray(DebuggingClassWriter.java:73)
at net.sf.cglib.core.DefaultGeneratorStrategy.generate(DefaultGeneratorStrategy.java:26)
at net.sf.cglib.core.AbstractClassGenerator.create(AbstractClassGenerator.java:216)
at net.sf.cglib.core.KeyFactory$Generator.create(KeyFactory.java:144)
at net.sf.cglib.core.KeyFactory.create(KeyFactory.java:116)
at net.sf.cglib.core.KeyFactory.create(KeyFactory.java:108)
at net.sf.cglib.core.KeyFactory.create(KeyFactory.java:104)
at net.sf.cglib.proxy.Enhancer.(Enhancer.java:69)
```

When the "IncompatibleClassChangeError" occurs, it is possible that:

- The static on the main method was forgotten.
- A legal class was used illegally.
- A class was changed and there are references to it from an another class by its old signatures. Try deleting all class files and recompiling everything.

Try these steps to resolve the "IncompatibleClassChangeError." (@javacodegeeks)

40. "FileNotFoundException"

This Java software error message is thrown when a file with the specified pathname does not exist.

```
@Override public ParcelFileDescriptor openFile(Uri uri, String mode) throws
FileNotFoundException {
    if (uri.toString().startsWith(FILE_PROVIDER_PREFIX)) {
        int m = ParcelFileDescriptor.MODE_READ_ONLY;
        if (mode.equalsIgnoreCase("rw")) m = ParcelFileDescriptor.MODE_READ_WRITE;
        File f = new File(uri.getPath());
        ParcelFileDescriptor pfd = ParcelFileDescriptor.open(f, m);
        return pfd;
    } else {
        throw new FileNotFoundException("Unsupported uri: " + uri.toString());
    }
}
```

In addition to files not existing the specified pathname, this could mean the existing file is inaccessible.

Read this discussion about why the "FileNotFoundException" could be thrown. (@StackOverflow)

41. "EOFException"

An "EOFException" is thrown when an end of file or end of stream has been reached unexpectedly during input. Here's an example from JavaBeat of an application that throws an EOFException:

```
import java.io.DataInputStream;
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
public class ExceptionExample {
    public void testMethod1() {
        File file = new File("test.txt");
        DataInputStream dataInputStream = null;
        try {
            dataInputStream = new DataInputStream(new FileInputStream(file));
            while (true) {
                dataInputStream.readInt();
            }
        } catch (EOFException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (dataInputStream != null) {
                    dataInputStream.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
        }
    }
    public static void main(String[] args) {
        ExceptionExample instance1 = new ExceptionExample();
        instance1.testMethod1();
    }
}
```

Running the program above results in the following exception:
java.io.EOFException
at java.io.DataInputStream.readInt(DataInputStream.java:392)
at logging.simple.ExceptionExample.testMethod1(ExceptionExample.java:16)
at logging.simple.ExceptionExample.main(ExceptionExample.java:36)

When there is no more data while the class DataInputStream is trying to read data in the stream, "EOFException" will be thrown. It can also occur in the ObjectInputStream and RandomAccessFile classes.
Read this discussion about when the "EOFException" can occur while running Java software. (@StackOverflow)

42. "UnsupportedEncodingException"
This Java software error message is thrown when character encoding is not supported (@Penn).
public UnsupportedEncodingException()

It is possible that the Java Virtual Machine being used doesn't support a given character set. Read this discussion of how to handle "UnsupportedEncodingException" while running Java software. (@StackOverflow)

43. "SocketException"
A "SocketException" indicates there is an error creating or accessing a socket (@ProgramCreek).

```
public void init(String contextName, ContextFactory factory) {
    super.init(contextName, factory);
    String periodStr = getAttribute(PERIOD_PROPERTY);
    if (periodStr != null) {
        int period = 0;
        try {
            period = Integer.parseInt(periodStr);
        } catch (NumberFormatException nfe) {}
        if (period <= 0) {
            throw new MetricsException("Invalid period: " + periodStr);
        }
        setPeriod(period);
    }
    metricsServers =
        Util.parse(getAttribute(SERVERS_PROPERTY), DEFAULT_PORT);
    unitsTable = getAttributeTable(UNITS_PROPERTY);
    slopeTable = getAttributeTable(SLOPE_PROPERTY);
    tmaxTable = getAttributeTable(TMAX_PROPERTY);
    dmaxTable = getAttributeTable(DMAX_PROPERTY);
```

```
    try {
        datagramSocket = new DatagramSocket();
    } catch (SocketException se) {
        se.printStackTrace();
    }
}
```

This exception usually is thrown when the maximum connections are reached due to:

- No more network ports available to the application.
- The system doesn't have enough memory to support new connections.

Read this discussion of how to resolve "SocketException" issues while running Java software. (@StackOverflow)

44. "SSLException"

This Java software error message occurs when there is failure in SSL-related operations. The following example is from Atlassian (@Atlassian):

```
com.sun.jersey.api.client.ClientHandlerException: javax.net.ssl.SSLException:
java.lang.RuntimeException: Unexpected error:
java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-
empty
    at
com.sun.jersey.client.apache.ApacheHttpClientHandler.handle(ApacheHttpClientHandler.jav
a:202)
    at com.sun.jersey.api.client.Client.handle(Client.java:365)
    at com.sun.jersey.api.client.WebResource.handle(WebResource.java:556)
    at com.sun.jersey.api.client.WebResource.get(WebResource.java:178)
    at
com.atlassian.plugins.client.service.product.ProductServiceClientImpl.getProductVersionsAft
erVersion(ProductServiceClientImpl.java:82)
    at com.atlassian.upm.pac.PacClientImpl.getProductUpgrades(PacClientImpl.java:111)
    at
com.atlassian.upm.rest.resources.ProductUpgradesResource.get(ProductUpgradesResourc
e.java:39)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelper$ResponseOutI
nvoker$1.invoke(DispatchProviderHelper.java:206)
    at
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelper$1.intercept(Dis
patchProviderHelper.java:90)
    at
com.atlassian.plugins.rest.common.interceptor.impl.DefaultMethodInvocation.invoke(Default
MethodInvocation.java:61)
    at
com.atlassian.plugins.rest.common.expand.interceptor.ExpandInterceptor.intercept(ExpandI
nterceptor.java:38)
```

at
com.atlassian.plugins.rest.common.interceptor.impl.DefaultMethodInvocation.invoke(DefaultMethodInvocation.java:61)
at
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelper.invokeMethodWithInterceptors(DispatchProviderHelper.java:98)
at
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelper.access$100(DispatchProviderHelper.java:28)
at
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelper$ResponseOutInvoker._dispatch(DispatchProviderHelper.java:202)
...
Caused by: javax.net.ssl.SSLException: java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty
...
Caused by: java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty
...
Caused by: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty

This can happen if:

- Certificates on the server or client have expired.
- Server port has been reset to another port.

Read this discussion of what can cause the "SSLException" error in Java software. (@StackOverflow)

45. "MissingResourceException"

A "MissingResourceException" occurs when a resource is missing. If the resource is in the correct classpath, this is usually because a properties file is not configured properly. Here's an example (@TIBCO):

java.util.MissingResourceException: Can't find bundle for base name localemsgs_en_US, locale en_US
java.util.ResourceBundle.throwMissingResourceException
java.util.ResourceBundle.getBundleImpl
java.util.ResourceBundle.getBundle
net.sf.jasperreports.engine.util.JRResourcesUtil.loadResourceBundle
net.sf.jasperreports.engine.util.JRResourcesUtil.loadResourceBundle

Read this discussion of how to fix "MissingResourceException" while running Java software.

46. "NoInitialContextException"

A "NoInitialContextException" occurs when the Java application wants to perform a naming operation but can't create a connection (@TheASF).

[java] Caused by: javax.naming.NoInitialContextException: Need to specify class name in environment or system property, or as an applet parameter, or in an application resource file: java.naming.factory.initial

[java] at javax.naming.spi.NamingManager.getInitialContext(NamingManager.java:645)
[java] at javax.naming.InitialContext.getDefaultInitCtx(InitialContext.java:247)
[java] at javax.naming.InitialContext.getURLOrDefaultInitCtx(InitialContext.java:284)
[java] at javax.naming.InitialContext.lookup(InitialContext.java:351)
[java] at org.apache.camel.impl.JndiRegistry.lookup(JndiRegistry.java:51)

This can be a complex problem to solve but here are some possible problems that cause the "NoInitialContextException" Java error message:

- The application may not have the proper credentials to make a connection.
- The code may not identify the implementation of JNDI needed.
- The InitialContext class may not be configured with the right properties.

Read this discussion of what "NoInitialContextException" means when running Java software. (@StackOverflow)

47. "NoSuchElementException"

A "NoSuchElementException" happens when an iteration (such as a "for" loop) tries to access the next element when there is none.

```
public class NoSuchElementExceptionDemo{
   public static void main(String args[]) {
      Hashtable sampleMap = new Hashtable();
      Enumeration enumeration = sampleMap.elements();
      enumeration.nextElement();  //java.util.NoSuchElementExcepiton here because
enumeration is empty
   }
}
```

Output:

```
Exception in thread "main" java.util.NoSuchElementException: Hashtable Enumerator
      at java.util.Hashtable$EmptyEnumerator.nextElement(Hashtable.java:1084)
      at test.ExceptionTest.main(NoSuchElementExceptionDemo.java:23)
```

The "NoSuchElementException" can be thrown by these methods:

- Enumeration::nextElement()
- NamingEnumeration::next()
- StringTokenizer::nextElement()
- Iterator::next()

Read this tutorial of how to fix "NoSuchElementException" in Java software. (@javinpaul)

48. "NoSuchFieldError"

This Java software error message is thrown when an application tries to access a field in an object but the specified field no longer exists in the onbject (@sourceforge).

public NoSuchFieldError()

Usually, this error is caught in the compiler but will be caught during runtime if a class definition has been changed between compile and running.

Read this discussion of how to find what causes the "NoSuchFieldError" when running Java software. @StackOverflow

49. "NumberFormatException"

This Java software error message occurs when the application tries to convert a string to a numeric type, but that the number is not a valid string of digits (@alvinalexander).

```
package com.devdaily.javasamples;
public class ConvertStringToNumber {
   public static void main(String[] args) {
      try {
         String s = "FOOBAR";
         int i = Integer.parseInt(s);
         // this line of code will never be reached
         System.out.println("int value = " + i);
      }
      catch (NumberFormatException nfe) {
         nfe.printStackTrace();
      }
   }
}
```

The can "NumberFormatException" be thrown when:

- Leading or trailing spaces in the number.
- The sign is not ahead of the number.
- The number has commas.
- Localisation may not categorize it as a valid number.
- The number is too big to fit in the numeric type.

Read this discussion of how to avoid "NumberFormatException" when running Java software. (@StackOverflow).

50. "TimeoutException"

This Java software error message occurs when a blocking operation times out.

```
private void queueObject(ComplexDataObject obj) throws TimeoutException,
InterruptedException {
   if (!queue.offer(obj, 10, TimeUnit.SECONDS)) {
      TimeoutException ex = new TimeoutException("Timed out waiting for parsed elements
to be processed. Aborting.");
      throw ex;
   }
}
```

Read this discussion about how to handle "TimeoutException" when running Java software. (@StackOverflow).

Conclusion

And that wraps it up! If you've followed along the whole way, you should be ready to handle a variety of runtime and compiler errors and exceptions. Feel free to keep both of these articles saved or otherwise bookmarked for quick recall. And for the ultimate Java developer's toolkit, don't forget to download The Comprehensive Java Developer's Guide.