Java 用語集（Java 講義資料下巻）

用語	内容	用途・使用方法
abstract 修飾子	抽象クラス、抽象メソッドを定義する際に使用する修飾子です。	修飾子 abstract class クラス名 {}
append()メソッド	オブジェクトの後ろに文字列を連結する際に使用するメソッドです。	StringBuilder sb = new StringBuilder(); sb.append(“あいうえお”);
catch ブロック	例外処理において try ブロックで例外が発生した場合に実行されるブロックです。 例外発生の処理を記述します。	try {} **catch() {}** finally {}
equals()メソッド	2 つの変数が同じオブジェクトを参照しているかを判定します。 特に String クラスの equals()メソッドは文字列の比較に使用されます。	String a = “abc”; String b = “xyz”; if (a.equals(b)) {}
Error クラス	エラーに関する例外クラスになり、下記が代表的な例外クラスです。	【代表的な例外クラス】 ・VirtualMachineError ・OutOfMemoryError ・StackOverflowError ・NoClassDefFoundError
Exception クラス	例外全般を扱う例外クラスです。	【RuntimeException 以外の代表的な例外クラス】 ・IOException ・FileNotFoundException ・ClassNotFoundException

用語	内容	用途・使用方法
extends	サブクラスを宣言する場合に「extends」という予約語の後ろにスーパークラスを指定します。	サブクラス名 **extends** スーパークラス名 {}
FileNotFoundException	ファイルやフォルダに対して不適切な操作を実行した際に発生する例外です。	
final	定数や処理内容を変更させたくないメソッドに使用します。	final String STR = “定数”; ※final を付けた際は定数とわかるように変数名を全て大文字にするのが一般的です。
finally ブロック	例外処理で例外の有無に関わらず必ず実行したい処理を finally ブロックに定義します。	try {} catch() {} **finally {}**
import 文	異なるパッケージ内のクラスにクラス名だけでアクセスできるようにするための宣言です。	import インポートしたいクラス;
indexOf()メソッド	引数の文字または文字列が最初に出現する位置を返します。	String str = “あいうえおかきくけこ”; int index = str.indexOf(“かき”); ⇒index は「5」になります。
instanceof 演算子	変数が参照するオブジェクトが何のクラスまたはインターフェイスかを判定します。	変数名 instanceof クラス名またはインターフェイス名 ⇒右辺に一致する型の場合、true が返ります。
Integer クラス	ラッパークラスの 1 つで、int 型に関する様々な機能を提供するクラスです。	機能の 1 つとして、下記は文字列を整数値に変換する例です。 String str = “123”; int number = Integer.parseInt(str);
interface	インターフェイスを定義する際に使用する予約語です。	修飾子 interface インターフェイス名 {}
IOException	入出力処理（ファイルが存在しない等）の失敗時に発生する例外です。	

用語	内容	用途・使用方法
java.io パッケージ	入出力関連のクラスがまとまったパッケージです。	【主なクラス例】 ・BufferedReader ・InputStreamReader
java.lang パッケージ	基本的なクラスがまとまったパッケージです。	【主なクラス例】 ・Integer ・String
java.math パッケージ	数値の演算子に関するクラスがまとまったパッケージです。	【主なクラス例】 ・BigDecimal
java.util パッケージ	ユーティリティ関連のクラスがまとまったパッケージです。	【主なクラス例】 ・ArrayList<E> ・Date
Math クラス	指数関数、対数関数、平方根、および三角関数といった基本的な数値処理を実行するための機能をまとめたクラスです。	【主な Math クラスのメソッド】 ・abs() ・cell() ・floor() ・max() ・min() ・random()
new 演算子	クラスは宣言しただけでは利用することができないため、「new 演算子」を使用してオブジェクトを生成します。	【オブジェクトの生成】 クラス名 変数名 = **new** クラス名();
null	null を変数に代入することにより、何も参照していない状態になります。	
Object クラス	全てのクラスの継承元にあたるクラスで、全てのクラスが暗黙的に継承しているものになり、Object クラスのメンバを呼び出したりオーバーライドすることができます。	【Object クラスの代表的なメソッド】 ・String toString() ・boolean equals(Object obj)

用語	内容	用途・使用方法
package 文	パッケージはクラスをカテゴリ別に管理するためのものになり、ソースファイルの先頭行にクラスが所属するパッケージ名を指定します。	【package の指定】 package パッケージ名;

private メンバ	private 修飾子が付与されたメンバを private メンバと呼び、他のクラスからアクセスすることができなくなります。	【private の記述例】
		public class PrivateTest {
		private int num;
		private String str;
		}

protected 修飾子	サブクラスからスーパークラスのメンバにアクセスしたい場合に指定する修飾子です。	【protected の記述例】
		public class ProtectedTest {
		protected String str;
		}

public メンバ	他の全てのクラス、メソッドからアクセス可能な public 修飾子が付与されたメンバです。	【public の記述例】
		public class PublicTest {
		public int num;
		}

return 文	メソッドに戻り値の型を指定している場合、メソッドの呼び出し元に戻り値を返す必要があり、戻り値を返す際に「return 文」を記述します。
	また、戻り値は 1 つの値（式）しか指定できません。
		【メソッドの定義】
		修飾子 戻り値の型 メソッド名(引数) {
		return 戻り値;
		}

用語	内容	用途・使用方法
RuntimeException クラス	全ての非チェック例外のスーパークラスです。	【代表的な例外クラス】 ・ClassCastException ・IllegalArgumentException ・NumberFormatException ・IllegalStateException ・IndexOutOfBoundsException ・ArrayIndexOutOfBoundsException ・NullPointerException ・ArithmeticException

static	オブジェクトではなく、クラス全体に関連付けられたメンバとして扱われ、クラス全体に関連付けられたフィールドを static 変数(クラス変数)、メソッドを static メソッド(クラスメソッド)と言います。	【static 変数の宣言】
		static 型名 変数名;
		【static メソッドの定義】
		static 戻り値の型 メソッド名(引数リスト) {
		処理;
		}
		【static メソッドの呼び出し】
		クラス名.static メソッド名

StringBuilder クラス	文字列操作を行うためのクラスで、文字列の結合、部分置換、検索等を行うことができます。 文字列を連結する場合、連結前の文字列をメモリ上に残さずに連結されるため、メモリに負荷をかけることなく文字列の連結を行うことができます。	【文字列結合の例】 String str = "あいうえお"; StringBuilder builder = new StringBuilder(str); builder.append(“かきくけこ”);

用語	内容	用途・使用方法
String クラス	文字列を扱う為に設計されたクラスで、文字列はこのクラスから生成されたオブジェクトとして扱われます。 また、文字列を操作するメソッドも定義されています。	【String クラスのメソッド（抜粋)】 ・boolean equals(Object anObject) ・int length() ・String substring(int from) ・int indexOf(String str)
super()	スーパークラスのコンストラクタを指す処理で、スーパークラスのコンストラクタが呼び出されます。	【super()構文】 **super**(引数リスト);
super	サブクラスからスーパークラスのメソッドを呼び出す際に使用します。	【super 構文】 **super**.メンバ名;
this()	同じクラスに定義された別のコンストラクタを呼びだす際に使用します。	【this()構文】 **this**(引数リスト);
throw	throw 文は意図的に例外を発生させる処理です。 引数の値や入力した文字列が想定外の値だったときに意図的にメソッドの処理を中断したいときに使用します。	【throw 構文】 **throw** 例外クラスのオブジェクト
Throwable クラス	例外クラス。Java 言語の全てのエラーと例外のスーパークラスです。サブクラスとして Error クラスと Exception クラスが提供されています。	
throws	例外が発生する可能性のあるメソッドの場合、呼び出し元へその例外の情報を渡すことを明示する必要があります。 throws 文を使用して記述することを「throws 宣言」と言います。	【throws 構文】 修飾子 戻り値の型 メソッド名(引数リスト）**throws** 例外クラス名 {}

用語	内容	用途・使用方法
toString()メソッド	数値型などを String 型の文字列に変換するためのメソッドです。	【println()メソッドでの説明】 println()メソッドでは「toString()」といった表記をしていませんが、Object クラスから継承した toString()メソッドが呼び出されています。 **System.out.println(123);**
try ブロック	try ブロックを例外が発生しそうな箇所を囲み、プログラムが強制的に中断しないようにします。	【try-catch-finally 構文】 **try** {} catch() {} finally {}
void	戻り値を持たないメソッドを示す際に記述します。	【void 例】 public **void** Test() {}
アクセス修飾子	フィールドやメソッドへのアクセス制御を行うための修飾子です。	【アクセス修飾子の種類】 ・public ・protected ・なし ・private ※上から順にアクセスできる制限が厳しくなります。
インスタンス	クラスを new 演算子によって実体化させて使用できるようにしたものをオブジェクトと呼んでおり、インスタンスとも呼びます。	

インスタンス変数	オブジェクトを経由してアクセス可能な変数です。	【インスタンス変数】
		public class Test {
		int num = 1;
		}

用語	内容	用途・使用方法

インスタンスメソッド	オブジェクトを経由してアクセス可能なメソッドです。	【インスタンスメソッド】
		public class Test {
		　public void show() {
		　　System.out.println(“Hello Java”);
		　}
		}

インターフェイス	オブジェクトを利用する側に公開すべき操作（メソッド）などをまとめたプログラムです。	【インターフェイス構文】 修飾子 interface インターフェイス名 {}
インポート	クラス名だけでアクセスするために記述する文です。	【インポート文】 import パッケージ名.クラス名;
オーバーライド	スーパークラスで定義されたメソッドと同じ名前のメソッドをサブクラスで再定義します。	

オーバーロード	同じクラスの中に同じ名前のメソッドを 2 つ以上定義することを指します。
	オーバーロードを行うには下記のいずれかの条件を満たしている必要があります。
	・各メソッドの引数の型が異なる。
	・各メソッドの引数の個数が異なる。
	・各メソッドの引数の順番が異なる。
		【オーバーロード例】
		public class Test {
		　public void show() {}
		　public void show(int num, String str) {}
		　public void show(String str, int num) {}
		}

オブジェクト	クラスを new 演算子によって実体化させて使用できるようにしたものをオブジェクトと呼んでおり、インスタンスとも呼びます。	

用語	内容	用途・使用方法
オブジェクト指向	効率よくシステム開発するための設計思想になり、下記の 3 つの重要な要素（オブジェクト指向の三大要素）があります。 ・カプセル化 ・継承 ・ポリモーフィズム（多態性）	
オブジェクト指向型言語	オブジェクト指向に沿った開発が行える言語仕様になっているプログラミング言語のことです。	【主なオブジェクト指向言語】 ・Java ・C++ ・C# ・Python ・PHP ・Visual Basic
ガーベッジコレクション	オブジェクトがどの変数からも扱われなくなった場合に Java の判断によりオブジェクトが破棄されます。	

カプセル化	フィールドとメソッドを 1 つのクラスにまとめ、保護したいメンバに private を付けることで外からのアクセスを制限する仕組みのことです。	【カプセル化例】
		public class Test {
		　private int num;
		　public void setNum(int num) {
		　　this.num = num;
		　}
		　public int getNum() {
		　　return num;
		　}
		}

用語	内容	用途・使用方法
仮引数	メソッドが引数を受け取るための変数のことです。	【仮引数例】 public void test(**仮引数**) {}
基底クラス	=スーパークラス	
基本型	データ型の 1 つで、基本型には「数値」、「文字」、「真偽値」があります。 プリミティブ型とも呼びます。	【基本型の種類】 ・short ・int ・double ・float ・long ・boolean ・char ・byte

具象クラス	具象メソッドだけを持つクラスのことです。
	また、抽象メソッドを持つクラスは「抽象クラス」です。
		【具象クラス構文】
		class クラス名 {
		・・・・・
		}

具象メソッド	具体的な処理が記述できるメソッドのことです。
	また、具体的な処理を記述しないメソッドは「抽象メソッド」です。
		【具象メソッド構文】
		戻り値の型 メソッド名(引数リスト) {
		・・・・・
		}

クラス	オブジェクトの「状態・性質」、「機能」をまとめたものです。	
クラス型	変数の型にクラス名を指定することです。	
クラス変数	static を付けたフィールドを「クラス変数」、メソッドを「クラスメソッド」と呼びます。	

用語	内容	用途・使用方法
クラスメソッド	static を付けたメソッド「クラスメソッド」、フィールドを「クラス変数」と呼びます。	
クラスライブラリ	汎用的な機能を持ったプログラムの集まりのことです。	
継承	メンバ（フィールドやメソッド）を他のクラスに引き継がせる機能のことです。	【継承構文】 class サブクラス名 **extends** スーパークラス名
コマンドライン引数	コマンドライン上でプログラムを実行する際に、クラス名に続けて半角スペース区切りで文字列を 1 つ以上記述でき、これらの文字列がコマンドライン引数です。	【「test」、「hello」、「123」がコマンドライン引数の例】 > java Sample test hello 123

コンストラクタ	オブジェクトが生成された直後に 1 度だけ実行される処理のことです。
	下記はメソッドとの違いです。
	・コンストラクタの名前はクラス名と同じ
	・戻り値の型を指定しない
		【コンストラクタ構文】
		修飾子 クラス名(引数リスト) {
		・・・・・
		}

サブインターフェイス	継承先のインターフェイスをサブインターフェイスと言います。	【サブインターフェイス構文】 **サブインターフェイス名** extends スーパーインターフェイス {}
サブクラス	継承先のクラスをサブクラスと言います。 スーパークラス内で定義されたメンバに対して、あたかも自分のクラスのメンバであるかのようにアクセスすることが可能です。	【サブクラス構文】 **サブクラス** extends スーパークラス {}
サブパッケージ	パッケージの中のパッケージのことをサブパッケージと呼び、親子関係の子にあたる部分です。	
参照型	基本データ型以外の、クラスや配列を変数の型に指定したものです。	

用語	内容	用途・使用方法
実引数	呼び出し元から渡される引数のことです。	【実引数構文】 メソッド名(**実引数**);
スーパーインターフェイス	継承元となるインターフェイスのことです。	【スーパーインターフェイス構文】 サブインターフェイス extends **スーパーインターフェイス** {}
スーパークラス	継承元となるクラスのことです。	【スーパークラス構文】 サブクラス extends **スーパークラス** {}
多重継承	2 つ以上のインターフェイスを実装することを言います。 ※Java では 2 つのクラスを継承することはできません。	【多重継承構文】 class クラス名 **implements インターフェイス名①, インターフェイス名②** {}
多態性	与えられた情報により、異なる振る舞いを行うオブジェクト指向型言語の特徴です。	
チェック例外	「コンパイラが発生を予測できる例外」であり、コンパイル時に発生し得るかどうかチェックされるものです。	
抽象クラス	処理内容を記述しないメソッドや、それを持つクラスを定義することです。 抽象クラスを継承して具象クラスを作成する場合、抽象メソッドをオーバーライドしないとコンパイルエラーが発生します。	【抽象クラスと抽象メソッド構文】 **修飾子 abstract class クラス名 {** **フィールド** **修飾子 abstract 戻り値の型 メソッド名(引数リスト);** **}**
抽象メソッド	処理内容を記述しないメソッドのことです。	【抽象メソッド構文】 「抽象クラス」参照
デフォルトコンストラクタ	クラスにコンストラクタを定義していない場合に用意されている「引数のないコンストラクタ」のことであり、自動的に実行されます。	

用語	内容	用途・使用方法
デフォルトパッケージ	パッケージを指定しない場合に属する暗黙的に決められたパッケージのことです。	
名前空間	パッケージのようにクラス名等の識別子を重複させないための仕組みのことです。	【同じクラス名を複数作成する場合】 パッケージ a の Test クラス パッケージ b の Test クラス
派生クラス	=サブクラス	
パッケージ	クラスをカテゴリ別に管理するものです。 クラスを機能や役割別に分類することでクラスを管理しやすくなります。	
引数	呼び出し先のメソッドに渡す値または、呼び出し元から受け取る値のことです	【引数構文（呼び出し先)】 戻り値の型 メソッド名(**引数**) {} 【引数構文（呼び出し元)】 メソッド名(**引数**);
非チェック例外	「コンパイラが発見できない例外」のことです。基本的に対処となる処理は記述不要です。	

フィールド	クラスのブロック内に宣言された変数のことです。	【フィールド構文】
		class Test {
		int num;
		String str;
		}

ポリモーフィズム	=多態性	
メソッド	1 つ以上の「処理」を 1 つの機能としてまとめたものです。	【メソッド構文】 **戻り値の型 メソッド名(引数リスト) {}**
メンバ	クラス内のフィールドとメソッドを合わせたものです。	

用語	内容	用途・使用方法

戻り値	メソッドの処理結果として呼び出し元のメソッドに渡す値のことです。	【戻り値構文】
		戻り値の型 メソッド名(引数) {
		return 戻り値;
		}

ラッパークラス	int や double などの基本型は自身の値を操作する機能などは持っていないため、基本型の値を保存して操作を行えるようにしたクラスものがラッパークラスです。	【ラッパークラス（抜粋)】 ・Byte ・Short ・Integer ・Long ・Float ・Double
乱数	ランダムな数値のことです。 Math クラスの random()メソッドで乱数を返すことができます。	【乱数メソッド】 static double random()
例外	プログラム実行中に発生した異常事態のことで、プログラムが続行できない異常事態のことを指します。 「例外」と「エラー」は別物で、例外は「プログラムの処理で対処できるもの」、エラーは「プログラム実行中に JVM では対処できない異常事態」のことです。	
例外クラス	例外の情報を扱っているクラスのことです。 その中でも標準的な例外クラスは、クラスライブラリとして提供されています。	【例外クラス】 ・Throwable クラス ・Error クラス ・Exception クラス ・RuntimeException クラス

用語	内容	用途・使用方法
例外処理	例外が発生した場合の対処となる処理を記述することです。 例外処理を記述しないと例外発生時にプログラムが強制的に中断されてしまいます。	

ローカル変数	メソッド内で宣言した変数のことです。
	メソッドの仮引数もローカル変数です。
		【ローカル変数の記述例】
		public class Test {
		public void show(**int num, String str**) {
		int x = 1;
		}
		}