Java

2018年12月6日 木曜日　　10:34

Javaの利点

1. OSを選ばない。WinとMacでもLinuxでも動作
2. 全てフリーソフトで実行環境を整えられる。
3. オブジェクト指向原語である。

入力する際の注意点

基本的には全て半角で入力

全角で入力できる場所は“” ‘’ の中だけ

コンパイラ

プログラム言語で記述されたファイルをバイトコードと呼ばる

Java VM用の言葉に一括翻訳する仕組み

インタプリタ

プログラム言語で記述されたファイルをマシンに対して通訳しながら

動作させる仕組み。JavaにおいてはJavaVMのことを指す。

定数をリテラルという

データ型の変数をカンマで区切って記述することで、複数の変数を一度に宣言することができる

整数は後ろに文字列が続くと文字列に変換される

ヒープ領域

プログラムが実行中にデータ量に合わせて変数領域を動的に確保するためのメモリ領域

イニシャライザは実体化される際にコンストラクタが処理されるのと同じタイミングで処理される

staticイニシャライザはstaticな変数やメソッドが生成されるタイミングと同じタイミングで、クラスが使用される前までに一度だけ実行される

親がメソッドでthrows Exceptionしていた場合は子クラスでオーバーライドする際は、同じ例外かサブクラス、または例外なしかの3パターンで再定義しなければならない

オーバーライドしたメソッドで親クラスの機能に関係ない例外を発生させるのはis一a関係から見てもあってはならない

ネストはあまりしない。2つぐらいがベスト
3項演算子も多用しない。

スレッド

2018年12月11日 火曜日　　13:46

マルチスレッド

複数のスレッドを作っても順番にじっこうされる。

交互に実行する場合はsleepやyieldを使う

sleepとyieldの違い

どちらもスレッドを一時停止させるが、sleepの方は実行されると無条件で一時停止を行う。このときに優先度の低いメソッドにも実行の機会が与えられる。

yieldは同じ優先順位の実行可能なスレッドが他になければ何も行わない

sleepはInterruptedExceptionの可能性があるので例外処理必須

スレッドの優先順位を設定するには、setPriority(int i)を使う

引数は1から10を設定。10が優先度最高。メインスレッドは5が設定

他のスレッドの終了を待つにはjoin()を使う。

一度最後まで実行されたスレッドのオブジェクトは二度とrun()の処理を行わない。

これをスレッドの消滅という。

IllegalThreadExceptionの例外が発生する

スレッドの作成

2018年12月11日 火曜日　　14:13

スレッドの作成は2通りある

1.Threadクラスを継承したサブクラスを作成

2.Runnableインタフェースを実装して作成するスレッド

1と2共にrun()をオーバーライド後、メインクラスでstart()を実行

1は記述が単純だが、Threadクラスを継承しているので、他のクラスを継承できない。
2は他のクラスを継承できるので、自由度が高い。

マルチスレッド下では、同じオブジェクトへアクセスするときはこちらの意に反した結果になる場合があるので注意

Synchronizedはメソッド、オブジェクトにロックをかけることができる。
スレッドの割り込みを防ぐことができる。
ロックのかかるメソッド、オブジェクトをモニタと呼ぶ。
モニタにアクセスすることをモニタを取得するという

スレッドの停止にはwait、再開にはnotifyを使用
Threadクラスのメソッドではなく、Objectクラスのメソッド
Synchronizedなオブジェクトの中でしか使えない

ThreadクラスのstaticメソッドとしてcurrentThread()メソッドは現在実行中のスレッドへの参照を取得することができる。
getName()はそのスレッドの名前を取得する

```
String obj;
synchronized (obj){　処理　}
```

上記のように、直接オブジェクトにロックをかけることも可能

スレッドまとめ

2018年12月11日 火曜日　　15:40

Wait()、notify()、notifyAll()メソッドはsynchronizedなオブジェクトの中でしか使用できない。
Wait()、notify()、notifyAll()メソッドを呼び出すスレッドは、そのオブジェクトに対するモニタを取得していなければならない。

Wait()メソッドは、オブジェクトのロックを開放します。waitしたスレッドは、notify(),notifyAll()メソッドで起こされても、notify(),notifyAll()メソッドを呼び出したスレッドがオブジェクトのロックを開放するまで動くことはない。

Javaの全てのプログラムはスレッドによって実行されている。
Javaでは、マルチスレッドの機能をサポートしている。
スレッドの生成方法にはjava.lang.Threadクラスを用いた方法とjava.lang.Runnableインタフェースを用いた方法がある。

sleepメソッドは一時停止を行い、他のスレッドに実行権を譲る
yieldメソッドは優先度の同じ実行可能なスレッドがある場合、一時停止をして実行権を譲る

joinメソッドはスレッドの終了を待つのに使用する
waitメソッドは別のスレッドからの通知を待つ際に使用し、
Notify(),notifyAll()によって呼び起こされるまで、実行停止している

あるオブジェクトに1つのスレッドしか入れないようにするには、synchronizedを使用する。Wait(), notifyAll(), notify()メソッドを使用する場合はこのsynchronizedなオブジェクトの中である必要があります。

StringBuilder

2018年12月11日 火曜日　　16:26

JavaSE5からStringBuilderが追加された。
StringBuilderとStringBufferはクラスが持つメソッドに違いはない。
StringBuilderは同期をとらないので、StringBufferより処理が早い。
複数の処理から文字列を変更するのであればStringBuffer、単一処理ならStringBuilderを使用

複数の処理から同時に変更が行われた際に必ず順番通りに変更が行われることを同期をとるという。
非同期では実行環境やタイミングによって実行結果は変わってしまう可能性がある。

Fileクラス

2018年12月11日 火曜日　　17:56

Fileクラスをインスタンス化してオブジェクトを生成する。
オブジェクトを生成する際に、実際にファイルを作成しているわけではなく、”hoge”というファイルへのパス(ファイル名を含む)を指定する。
プログラムがいずれかの場所にあるファイルにアクセスする土台をFileクラスは担う。
ファイルに対して入出力を行う場合、Fileクラスを使うケースは非常に多い

Fileオブジェクトの参照を引数に渡して、FlleWritterクラスをインスタンス化する。
書き込みはwrite()を使用。改行は¥n。実際にファイルに書き出すにはflush()を使用。
Write()は書き込む文字列を指定しただけなので実際に書き込む場合は必ずflush()を使用

Fileオブジェクトの参照を引数に渡して、FileReaderクラスをインスタンス化する。
Read()は読み込んだファイルの中を全て読み取って引数で渡されたchar配列に1文字ずつデータを入れる。読み取ったデータのバイト数をint型の戻り値で返す。

BufferdReaderとWriter

2018年12月11日 火曜日　18:08

BufferedWriterクラス

FileWriterクラスオブジェクトを参照している変数を引数に受け取り、BufferedWriterクラスをインスタンス化する。
BufferedWriterクラスは、文字をバッファリング(一時的にデータを溜め込む)することにより文字、配列、文字列を効率よく文字列型出力ストリームに書き込みできる。

Writer()を使用して、ファイルに書き出したいデータをStringクラスで指定する。
Fileクラスは改行の際に¥nが必要だが、BufferedWriterではnewLine()を呼ぶだけで改行することができる。

BufferedReaderクラス

FileWriterクラスオブジェクトを参照している変数を引数に受け取り、BufferedReaderクラスをインスタンス化する。1文字ずつではなく、行単位で処理が行える。
BufferedReaderクラスは、文字をバッファリング(一時的にデータを溜め込む)することにより文字、配列、文字列を効率よく文字型入力ストリームに読み込みできる。

readLine()はファイルの中のテキストデータを1行単位で取得してStringクラスで返すメソッド
テキストデータが何もない場合はnullを返す。
通常、テキストファイルの読み込み、書き込みはこの方法を使うので扱えるようにする。

読み込みの場合でもストリームを繋いでおくとメモリを確保してしまうので、必要がなくなったらclose()を呼んでストリームを切断する。
flush()を使用し、最後にclose()するのを忘れないようにする。

PrintWriterクラス

2018年12月12日 水曜日　　10:18

JavaSE5で機能拡張された。
FileWriterやBuffered Writerをラッピングしないと使用できなかったが、SE5からは使用しなくてもよくなった。

Fileクラス型のオブジェクトへの参照をもらった状態でPrintWriterクラスをインスタンス化する。
println()を使用して、文字列をバッファ内に書き込むと同時に改行する。
その後、flush()を使用し、最後にclose()する。

コレクション

2018年12月12日 水曜日　10:27

複数のデータをまとめて扱うためのクラスの総称をコレクションと呼ぶ。
格納できるのは、参照型のみで基本データ型は格納できない。
基本データ型を扱うには、ラッパークラスを使用する。配列も格納可能。
要素が参照型の場合は、格納されるのはオブジェクトへの参照(アドレス)。

コレクションインタフェース

Collection-List	要素をインデックスと関連づけて管理するため、同じ要素を重複登録可能
-Set	登録されたエレメントを一意に識別するため、同じ要素の重複不可
Map	要素をキー値とペアとして管理する。キーにはオブジェクトを用いる。 要素の重複登録は可能だが、キーの重複は不許可。

List	順序	ソート状態	同期
ArrayList	インデックス	不可	
Vector	インデックス	不可	○
LinkedList	インデックス	不可	
Set			
HashSet	なし	不可	
LinkedHashSet	挿入順	不可	
TreeSet	ソート済み	自然順序/独自の比較規則	
Map			
HashMap	なし	不可	
Hashtable	なし	不可	○
TreeMap	ソート済み	自然順序/独自の比較規則	
LinkedHashMap	挿入順	不可	

コレクションで型を指定しないと、警告が出る。実行自体は可能。
@SuppressWarnings(“unchecked”)を使用するとコンパイル時のエラーを回避できる。

Listインタフェース

2018年12月12日 水曜日　10:37

データ間に明らかな前後関係が存在する場合はListを利用する。

ArrayList

サイズが拡張できる配列のようなもの。参照の処理速度を要求されるときに適している

ソートはインデックスによる順次付きコレクションにかかわらず実装されていない。
挿入や、削除などの変更処理が多い場合はLinkedListの方が向いている。

要素としてnullを追加できる。

マルチスレッド環境で、複数のスレッドが同時のコレクションの内容を変更した時に、

内容は保障されないので、プログラマが同期を行う必要がある。
同期を行わない点を除いてVectorとほぼ同じ機能を提供。

ArrayListは同期を行わない分高速

LinkedList

要素同士が双方向にリンク(次要素と前要素の参照を保持)した構造になっている

要素の追加や削除の処理の際に、関連する前後の要素の参照を組み替えるだけで済むの

で、追加や削除の処理が多い処理に適している。

LinkedListはスタックやキューの実装に向いている。

Vector

基本的な機能はArrayListと同じだが、Vectorの一連のメソッドはスレッドセーフ(複数の

スレッドから呼ばれても問題が生じない作りになっている)のための同期化がされている
そのため、複数のスレッドから同時にアクセスできない。同期化すると実行速度が低下するの

で、特別な理由がない限りArrayListを使用した方が良い。

Vectorは同期化されること以外はArrayListとほぼ同等。

Setインタフェース

2018年12月12日 水曜日　　10:38

コレクションに重複する値を格納できないという特別な条件がある場合はSetを使用
重複不可。重複の判定にはequals()によって判定される。
null値を格納可能だが、1つのセットで1つだけnull値が強化される。

HashSet

ソートできないセット。オブジェクトの格納時にハッシュコード(オブジェクトを数値で表現したもの)を使用し、同じオブジェクト仮装でないかを判断するのでハッシュコードを生成するhashCode()の効率を高めれば、要素へのアクセス速度も向上する。要素の順序は一定には保たれず、同期化も行わない。
nullを要素として追加可能。重複を認めない場合はHashSetを使用する。

LinkedList

HashSetに双方向のリンク機能をつけたもの
要素を順序づけして、取り出したい時に適している。
この順序は要素がセットに挿入された順序になる(挿入順)

TreeSet

要素の昇順でソートされる。順序は自然順序(アルファベット、数値順など)になる
独自の比較規則を設定することも可能。同期化は行われない。

Setインタフェースは重複項目を持たないので、equals()で比較してもtrueが帰る要素ペアを持たない。

Mapインタフェース

2018年12月12日 水曜日　　10:40

任意のキー値を使って格納される要素の参照を行う場合はMapを使用
キー値は、マップに格納されている要素を探すための目印になるオブジェクト
Mapでは要素に一意にのキー値を割り当てするが、キーと値はどちらもオブジェクト。

キーがマップのどの場所に格納されているかはキー値のハッシュコードに左右されるので、ハッシュコードを取得できるhashCode()の効率を高めれば、要素へのアクセス速度も向上する。HashSetも同様。
キー値はマップの中で一意の値でなくてはならない。

HashMap

シンプルなマップで順序を保持せず、同期化も行わない。キー値としてnullをひとつだけ持つことができる。重複キーは不許可。値のnullは重複可

HashTable

HashMapの機能を同期化したもの。キー値にnullは認められていない。

LinkedHashMap

LinkedHashSetのようにHashMapに双方向のリンク機能をつけたもの。双方向リンクの処理を行うために追加や削除の速度は多少遅くなるが、その分読み取りの処理速度は向上する。

TreeMap

MapのサブインタフェースであるsortedSetの機能を実装しているので、要素の自然順序に従ってソートされる。重複したキーは不許可で、同期化も行われない。TreeSetと同じくオブジェクトの作成時にコンストラクタで絵独自の比較規則を渡すことが可能。

ジェネリクス

2018年12月12日 水曜日　　12:10

Genericsとは、クラス・インタフェース・メソッドなどの型をパラメータとして定義することを可能にしたもの。

クラスの定義内で用いられる型を変数化し、インスタンス化しオブジェクトを生成するときまで扱う型を抽象的に表現しておき、生成時に<>の間に具体的な型名を指定することで汎用的なクラスやメソッドを特定の型に対応づけることができる。

コレクションでジェネリクスを用いると以下の利点がある

- コレクションから要素を取り出す際にキャストが不要になる
- 誤ったキャストをコンパイル時に発見することができる
- 予期せぬ型を要素として追加できないようにする

アサーション

2018年12月12日 水曜日　　12:29

アサーションとはプログラムが仕様通りになっているか、チェックするための機能。
条件が成り立つはずの場所にアサーションチェック用のコードを入れた場合、条件が偽であった場合には、アサーションエラー(例外)が発生する。

assert 条件式1;
assert 条件式1 : 式2;

条件式1は条件がBoolean値になるようなものでないといけない。
式2は値を返さないものを記述することはできない

コンパイル後、 java -ea で有効
デフォルトは無効。明示的に無効化する場合は java -da

全体のアサーションは無効にするが、特定のクラスは有効にする。
全体を有効にし、特定クラスのみ無効化することも可能。

オプションを複数つけた場合、基本的には先に記述したものよりも、後に記述したものが優先される。クラス名を指定するような詳細レベルの設定があれば、それが優先される。
Java -da:特定のクラス名 -ea 実行クラス名
上記の場合では、特定クラスのアサーションのみが無効になる。

アサーションは、あくまでデバッグ用の機能であることに気をつける

クラスファイルにアサーションが残るので削除するには、ソースに以下を記述
Static final Boolean asserts = false;
アサーションをクラスファイルに含める場合はtrueを、含めたくない時はfalseを指定。
その上で、アサーション式をメソッド内に記述する際には、フィールド変数assertsがtrueの場合のみアサーション式が実行されるように記述する。

効果的な使い方

2018年12月12日 水曜日　　15:09

アサーションの効果的な使い方

不適切な使用

1. publicメソッド内の引数のチェック
2. 副作用があるアサーション式の利用
3. ユーザからの入力処理
4. AssersionErrorをcatch句で受け取る

1.他にも例外が発生し、どの理由で例外が発生したか判断しにくい
2.アサーションが有効・無効のときで処理が異なる使い方をすることはできない
3.値を制御できないユーザからの入力には対応できない
4.アサーションはデバッグ用であり、最終的にエラーを消滅させることが目的

適切な使用

1.privateメソッド内の引数チェック
　　プログラマが引数を制御可能なため、アサーションを使用するのは適切

2.publicメソッド内でも絶対に実行されないと仮定される場合
　　処理が実行されるはずがないと思われる場所に、assert false; を記述すると、
　　その過程が成り立たなくなった場合にAssertionErrorが発生する。

メモリ領域の解放

2018年12月12日 水曜日　　15:19

ガーベッジコレクションはJVMが不要になったインスタンスを自動的に回収し、メモリを解放する機能。ガーベッジになるのはインスタンス。
new演算子でインスタンスかするとヒープと呼ばれるメモリ領域に確保される。
このインスタンスの参照が切れると、インスタンスはガーベッジになる。
JavaVMのタイムスケジューラでガーベッジの破棄するタイミングは管理されている。
プログラマが明示的に調整することはできないが、Systemクラスのgc()によってその実行を促すことはできる。また、インスタンスが回収された時、自動的にそのインスタンスのfinalize()が呼び出しされる。

可変長引数

2018年12月12日 水曜日　　15:33

メソッドに任意の数の引数を渡す場合、配列を用意し値を渡す方法の他に、可変長引数を使用することにより、配列をわざわざ作らなくても可変長の値を渡すことができる
E.g. void method(int... var)

今までのメソッドの定義と可変長引数のメソッドの定義がオーバーロードされている場合明示的に引数の個数がしてされている方が優先される。
今までのメソッドの定義と可変長引数のメソッドの定義をオーバーライドで使用も可能