The Real World "Referential Transparency"

参照透過の実際

吉村 優（Yoshimura Hikaru）

yyu@mental.poker

June 9, 2024 @ ScalaMatsuri 2024

https://github.com/y-yu/referential-transparency-slide（3127a75）

目次

参照透過

- 参照透過（*Referential transparency*）とは、式の構成要素がすべて同じなら、式の値は常に同じになる [1]

参照透過

- 参照透過（*Referential transparency*）とは、式の構成要素がすべて同じなら、式の値は常に同じになる [1]
- たとえば下記のコードは参照透過になる

```
val two = 1 + 1
two
two
```

```
1 + 1
1 + 1
```

図 1: 参照透過なコード例

参照透過

- 一方で下記のコードは参照透過**ではない**

```
val hello = {
  println("hello")
  "hello"
}
hello
hello
```

```
{
  println("hello")
  "hello"
}
{
  println("hello")
  "hello"
}
```

参照透過

- 一方で下記のコードは参照透過**ではない**

```
val hello = {
  println("hello")
  "hello"
}
hello
hello
```

```
{
  println("hello")
  "hello"
}
{
  println("hello")
  "hello"
}
```

- この左のコードは `hello`の出力が 1 回だが、一方で右のコードは 2 回出力される

Scala での参照透過

- 今みた `println`のように、入出力は参照透過ではない
  - 他にも時計へのアクセスとか

Scala での参照透過

- 今みた `println`のように、入出力は参照透過ではない
  - 他にも時計へのアクセスとか
- `Future`もインスタンシエイト時にスレッドが走ってしまうので参照透過ではない

```
val f1 = Future(/* なにか */)
val f2 = Future(/* なにか */)
f1.flatMap(_ => f2)
```

```
Future(/* なにか */)
  .flatMap(_ => Future(/* なにか */))
```

  - この☝ コードは異なる意味になる

Scala での参照透過

- 一方で Monix の Task[2] は参照透過

Scala での参照透過

- 一方で Monix の Task[2] は参照透過
- Extensible Effects（Eff）も参照透過をやっていくことが前提（？）

このディスカッションの内容

このディスカッションの内容

参照透過だと何がいいのか？

このディスカッションの内容

参照透過だと何がいいのか？

参照透過どれくらいがんばるのか？

このディスカッションの内容

参照透過だと何がいいのか？

参照透過どれくらいがんばるのか？

参照透過は分かりやすいか分かりにくいか？

このディスカッションの内容

参照透過だと何がいいのか？

参照透過どれくらいがんばるのか？

参照透過は分かりやすいか分かりにくいか？

参照透過は他の言語でどうか？

登壇者

Twitter @kmizu
GitHub kmizu

Twitter @kory__3
GitHub kory33

Twitter @_yyu_
GitHub y-yu

あと他にもだれか（？）

このディスカッションの内容

このディスカッションの内容

参照透過だと何がいいのか？

このディスカッションの内容

参照透過だと何がいいのか？

参照透過どれくらいがんばるのか？

このディスカッションの内容

参照透過だと何がいいのか？

参照透過どれくらいがんばるのか？

参照透過は分かりやすいか分かりにくいか？

このディスカッションの内容

参照透過だと何がいいのか？

参照透過どれくらいがんばるのか？

参照透過は分かりやすいか分かりにくいか？

参照透過は他の言語でどうか？

参考文献

[1] プログラミング言語論 ドキュメント Â» 関数型言語.
http://web.sfc.keio.ac.jp/~hattori/prog-theory/ja/functional.html#id3.
Accessed: 2024-06-08.

[2] Monix Task.
https://monix.io/docs/current/eval/task.html.
Accessed: 2024-06-08.

Thank you for the attention!