CS320

Identifiers

Sukyoung Ryu

March 6, 2019

Don'ts

- Don't import any other libraries.
- Don't use the `break` statement.
- Don't use the `while` and `for` loops.
- Don't use mutation.

If your homework does any of the above,
it will get 0 point.

Identifiers

The “Are the following two programs equivalent?” game

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = x + 1
f(10)
```

```
def f(y:Int):Int = y + 1
f(10)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = x + 1
f(10)
```

```
def f(y:Int):Int = y + 1
f(10)
```

YES

parameter is consistently renamed

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = x + 1
f(10)
```

```
def f(x:Int):Int = y + 1
f(10)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = x + 1
f(10)
```

```
def f(x:Int):Int = y + 1
f(10)
```

NO

not a use of the parameter anymore

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = x + 1
f(10)
```

```
def f(y:Int):Int = x + 1
f(10)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = x + 1
f(10)
```

```
def f(y:Int):Int = x + 1
f(10)
```

NO

not a use of the parameter anymore

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = y + 1
f(10)
```

```
def f(z:Int):Int = y + 1
f(10)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = y + 1
f(10)
```

```
def f(z:Int):Int = y + 1
f(10)
```

YES

parameter never used, so almost any name is ok

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = y + 1
f(10)
```

```
def f(y:Int):Int = y + 1
f(10)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = y + 1
f(10)
```

```
def f(y:Int):Int = y + 1
f(10)
```

NO

now a use of the parameter

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = y + 1
f(10)
```

```
def f(x:Int):Int = z + 1
f(10)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = y + 1
f(10)
```

```
def f(x:Int):Int = z + 1
f(10)
```

NO

still an undefined identifier, but a different one

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(z:Int):Int = {
  val y = 10
  z + y
}
f(0)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(z:Int):Int = {
  val y = 10
  z + y
}
f(0)
```

YES

parameter is consistently renamed

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(x:Int):Int = {
  val z = 10
  x + z
}
f(0)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(x:Int):Int = {
  val z = 10
  x + z
}
f(0)
```

YES

local identifier is consistently renamed

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(x:Int):Int = {
  val x = 10
  x + x
}
f(0)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(x:Int):Int = {
  val x = 10
  x + x
}
f(0)
```

NO

local identifier now hides the parameter

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(y:Int):Int = {
  val y = 10
  y + y
}
f(0)
```

Identifiers

"Are the following two programs equivalent?"

```
def f(x:Int):Int = {
  val y = 10
  x + y
}
f(0)
```

```
def f(y:Int):Int = {
  val y = 10
  y + y
}
f(0)
```

NO

local identifier now hides the parameter

Free and Bound Identifiers

An identifier for the parameter of a function or the name of a local identifier is a *binding occurrence*

```
def f(x, y) = x + y + z

def g() = {
  val a = 3
  val c = 4
  a + b + c
}
```

Free and Bound Identifiers

A use of a function parameter or a local identifier is a *bound occurrence*

```
def f(x, y) = x + y + z

def g() = {
  val a = 3
  val c = 4
  a + b + c
}
```

Free and Bound Identifiers

A use of an identifier that is not a function parameter or a local identifier is a *free identifier*

```
def f(x, y) = x + y + z

def g() = {
  val a = 3
  val c = 4
  a + b + c
}
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{with {x {+ 1 2}}
  {+ x x}}
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{with {x {+ 1 2}}
  {+ x x}}
```
$\Rightarrow$ 6

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
x
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

`x` $\Rightarrow$ error: free identifier

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{+ {with {x {+ 1 2}}
     {+ x x}}
   {with {x {- 4 3}}
     {+ x x}}}
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{+ {with {x {+ 1 2}}
     {+ x x}}
   {with {x {- 4 3}}
     {+ x x}}}
```

$\Rightarrow$ 8

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{+ {with {x {+ 1 2}}
     {+ x x}}
   {with {y {- 4 3}}
     {+ y y}}}
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{+ {with {x {+ 1 2}}
     {+ x x}}
   {with {y {- 4 3}}
     {+ y y}}}          ⇒  8
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{with {x {+ 1 2}}
   {with {x {- 4 3}}
     {+ x x}}}
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{with {x {+ 1 2}}
   {with {x {- 4 3}}
     {+ x x}}}
```
$\Rightarrow$ 2

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{with {x {+ 1 2}}
   {with {y {- 4 3}}
     {+ x x}}}
```

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
{with {x {+ 1 2}}
   {with {y {- 4 3}}
     {+ x x}}}
```

$\Rightarrow$ 6

Arithmetic Expressions with Identifiers

```
<WAE> ::= <num>
        | {+ <WAE> <WAE>}
        | {- <WAE> <WAE>}
        | {with {<id> <WAE>} <WAE>}
        | <id>
```

```
trait WAE
case class Num(n: Int) extends WAE
case class Add(l: WAE, r: WAE) extends WAE
case class Sub(l: WAE, r: WAE) extends WAE
case class With(x: String, i: WAE, b: WAE) extends WAE
case class Id(x: String) extends WAE
```

Parser for AE with Identifiers

```
// parser for WAE
object WAE extends ExprParsers {
  lazy val wae: Parser[WAE] =
    int                          ^^ { case n     => Num(n)    } |
    wrap("+" ~> wae ~ wae) ^^ { case l ~ r => Add(l, r) } |
    wrap("-" ~> wae ~ wae) ^^ { case l ~ r => Sub(l, r) } |
    wrap("with" ~> wrap(str ~ wae) ~ wae) ^^ {
                          case x ~ i ~ b => With(x, i, b) } |
    str                          ^^ { case x => Id(x) }
  def apply(str: String): WAE =
    parse(wae, str).getOrElse(error(s"bad syntax: $str"))
}
```

```
Add(Sub(Num(3), Num(4)), Num(7))
Sub(Num(5), Num(1), Num(2))
With("x", Num(3), Add(Id("x"), Num(3)))
```

Exercise #1

- Available from the course webpage

  `http://plrg.kaist.ac.kr/doku.php?id=home:lectures:cs320_2019_1`

Sukyoung Ryu
sryu.cs@kaist.ac.kr
http://plrg.kaist.ac.kr