Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle       ::
  intersection ::
  trans        ::
  invert       ::
  scale        ::
  def          ::
  within       ::
  and          ::
  inv          ::
  lit          ::
```

?

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```

Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle       :: v Shape
  intersection :: (v Shape) (v Shape) -> v Shape
  trans        :: (v Point) (v Shape) -> v Shape
  invert       ::           (v Shape) -> v Shape
  scale        :: (v Real)  (v Shape) -> v Shape
  def          ::
  within       ::
  and          ::
  inv          ::
  lit          ::
```

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```

Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle       :: v Shape
  intersection :: (v Shape) (v Shape) -> v Shape
  trans        :: (v Point) (v Shape) -> v Shape
  invert       ::           (v Shape) -> v Shape
  scale        :: (v Real)  (v Shape) -> v Shape
  def          :: ((v a) -> In (v a) (v b)) -> v b
  within       :: (v Point) (v Shape) -> v Bool
  and          ::
  inv          ::
  lit          ::
```

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```

Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle       :: v Shape
  product      :: (v a) (v a) -> v a | * a
  trans        :: (v Point) (v Shape) -> v Shape
  invert       ::           (v Shape) -> v Shape
  scale        :: (v Real)  (v Shape) -> v Shape
  def          :: ((v a) -> In (v a) (v b)) -> v b
  within       :: (v Point) (v Shape) -> v Bool

  inv          ::
  lit          ::
```

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```

Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle       :: v Shape
  product      :: (v a) (v a) -> v a | * a
  trans        :: (v Point) (v Shape) -> v Shape
  invert       ::       (v a) -> v a | ~ a
  scale        :: (v Real)  (v Shape) -> v Shape
  def          :: ((v a) -> In (v a) (v b)) -> v b
  within       :: (v Point) (v Shape) -> v Bool


  lit          ::
```

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```

Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle        :: v Shape
  product       :: (v a) (v a) -> v a | * a
  trans         :: (v Point) (v Shape) -> v Shape
  invert        ::       (v a) -> v a | ~ a
  scale         :: (v Real)  (v Shape) -> v Shape
  def           :: ((v a) -> In (v a) (v b)) -> v b
  within        :: (v Point) (v Shape) -> v Bool


  lit           :: a -> v a | toString a
```

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```

Assignment 10: class-based embedding

grammar

```
E
 ::= Circle
 |   Intersection E E
 |   Trans Point E
 |   Invert E
 |   Scale Real E
 |   let Var = E in E
 |   inside Point E
 |   E * E
 |   ~ E
 |   Point
```

embedding in shapeStart.icl

```
:: Shape
 = Circle
 | Intersection Shape Shape
 | Trans  Point Shape
 | Invert       Shape
 | Scale  Real  Shape
let-binding
inside :: Point Shape -> Bool
instance * Shape, Bool
instance ~ Shape, Bool
:: Point = {x::Real, y::Real}
```

class-based embedding

```
class shape v where
  circle       :: v Shape
  product      :: (v a) (v a) -> v a | * a
  trans        :: (v Point) (v Shape) -> v Shape
  invert       ::       (v a) -> v a | ~ a
  scale        :: (v Real)  (v Shape) -> v Shape
  def          :: ((v a) -> In (v a) (v b)) -> v b
  within       :: (v Point) (v Shape) -> v Bool
  lit          :: a -> v a | toString a
```

```
e_plain =
    let p0   = {x=6.0, y=0.0} in
    let p1   = {x=3.0, y=0.0} in
    let disc = Scale 5.0 Circle in
    let lens = disc * (Trans p1 disc) in
    ~ (inside p0 lens) * inside p0 (~ lens)
```