Syllabus:

>> Challenge 1. Add a class

>> Challenge 2. Change the code to follow the standard

>> Challenge 3. Add a calculated static variable and a static constructor

>> Challenge 4. Add an instance variable Id and a constructor

>> Challenge 5 y 6. Add more similar classes

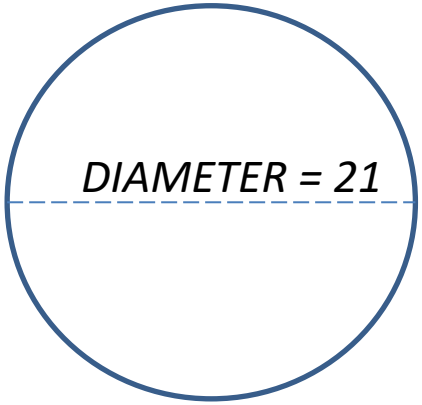
>> Challenge 7. Add the abstract Ball class

>> Challenge 8. Create DUMMY objects from the different classes

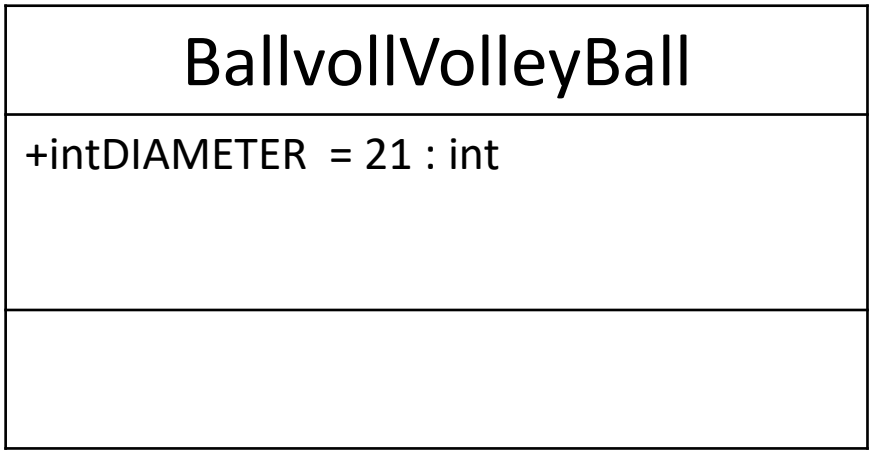
>> Challenge 9. Create an object factory for objects

>> Challenge 10. Add nullable and enum type variables

Real world

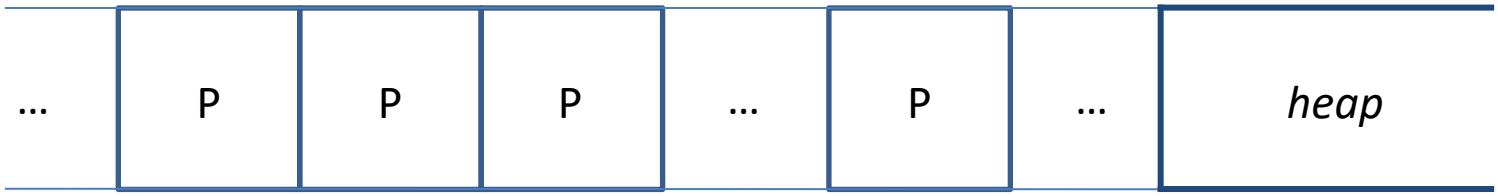


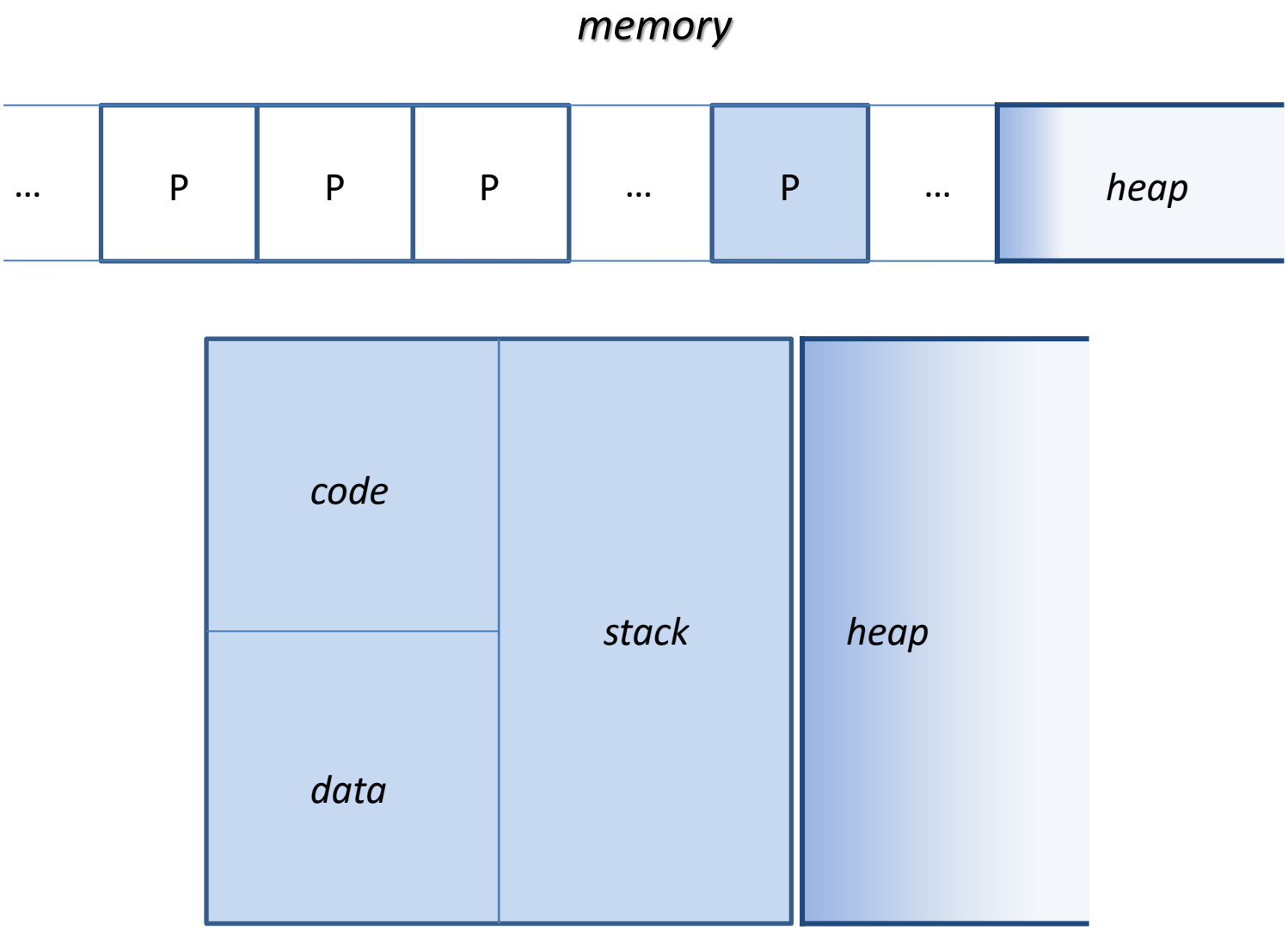
>> Challenge 1. Add a VolleyBall class

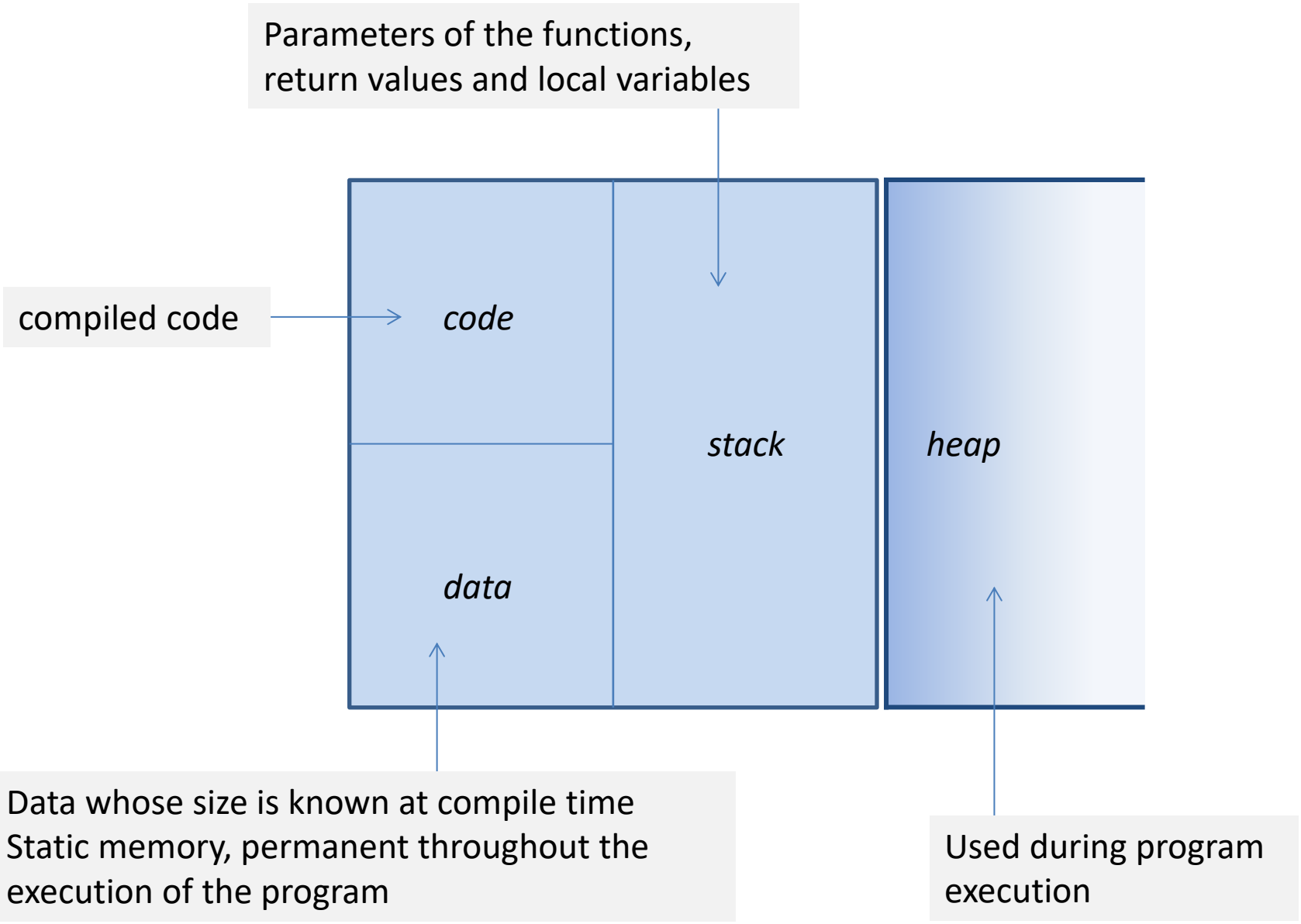


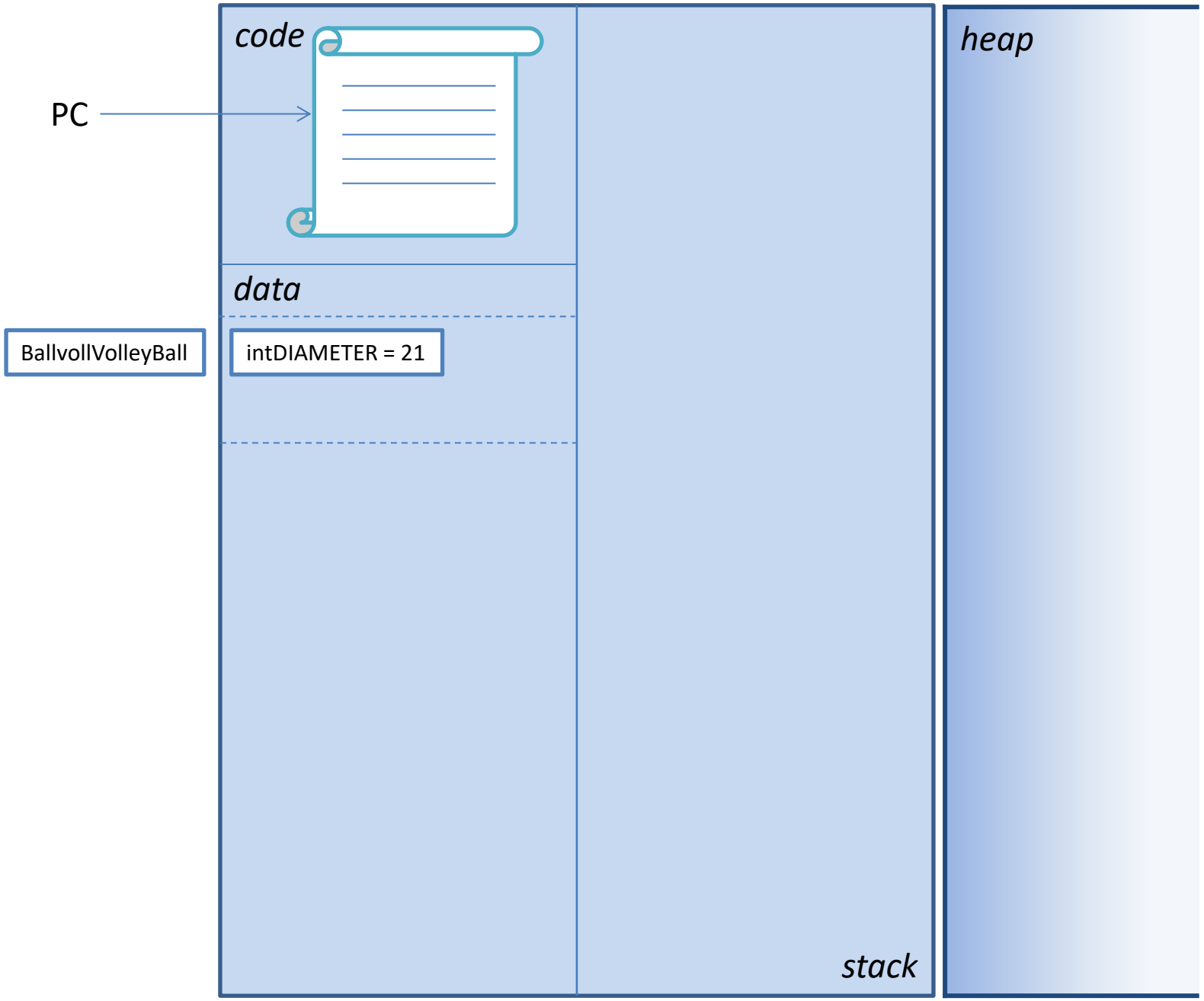
>> Challenge 2. Change the code to follow the standard

memory

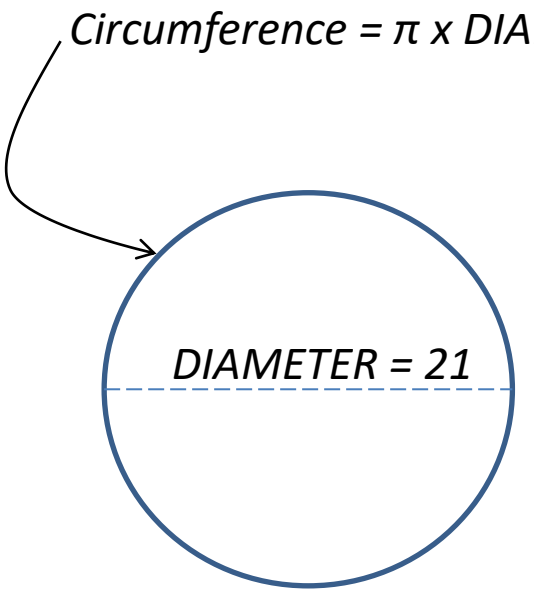



memory
...
P
P
P
...
P
...
heap
code
data
stack
heap

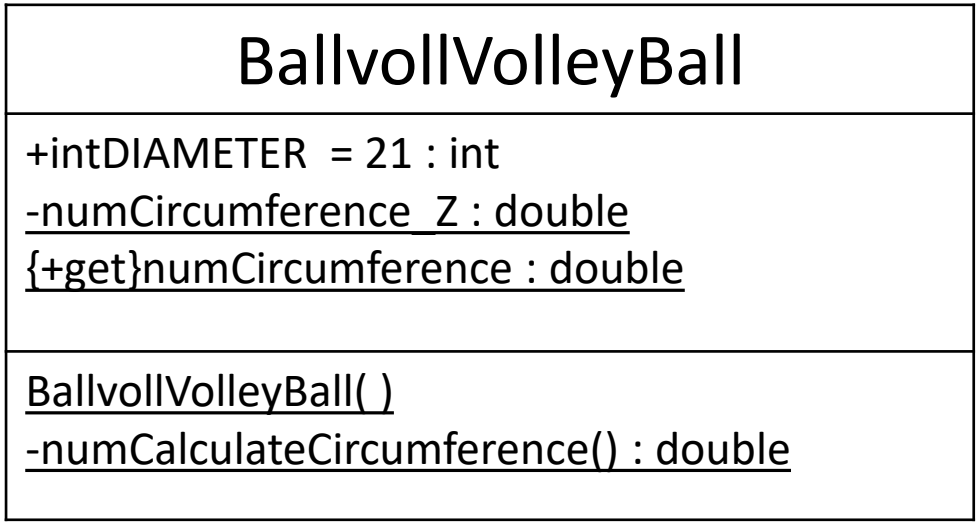

Parameters of the functions,
return values and local variables
compiled code
code
stack
heap
data
Data whose size is known at compile time
Static memory, permanent throughout the
execution of the program
Used during program
execution


code
PC
data
BallvollVolleyBall
intDIAMETER = 21
heap
stack

Real world



>> Challenge 3. Add a calculated static variable and a static constructor

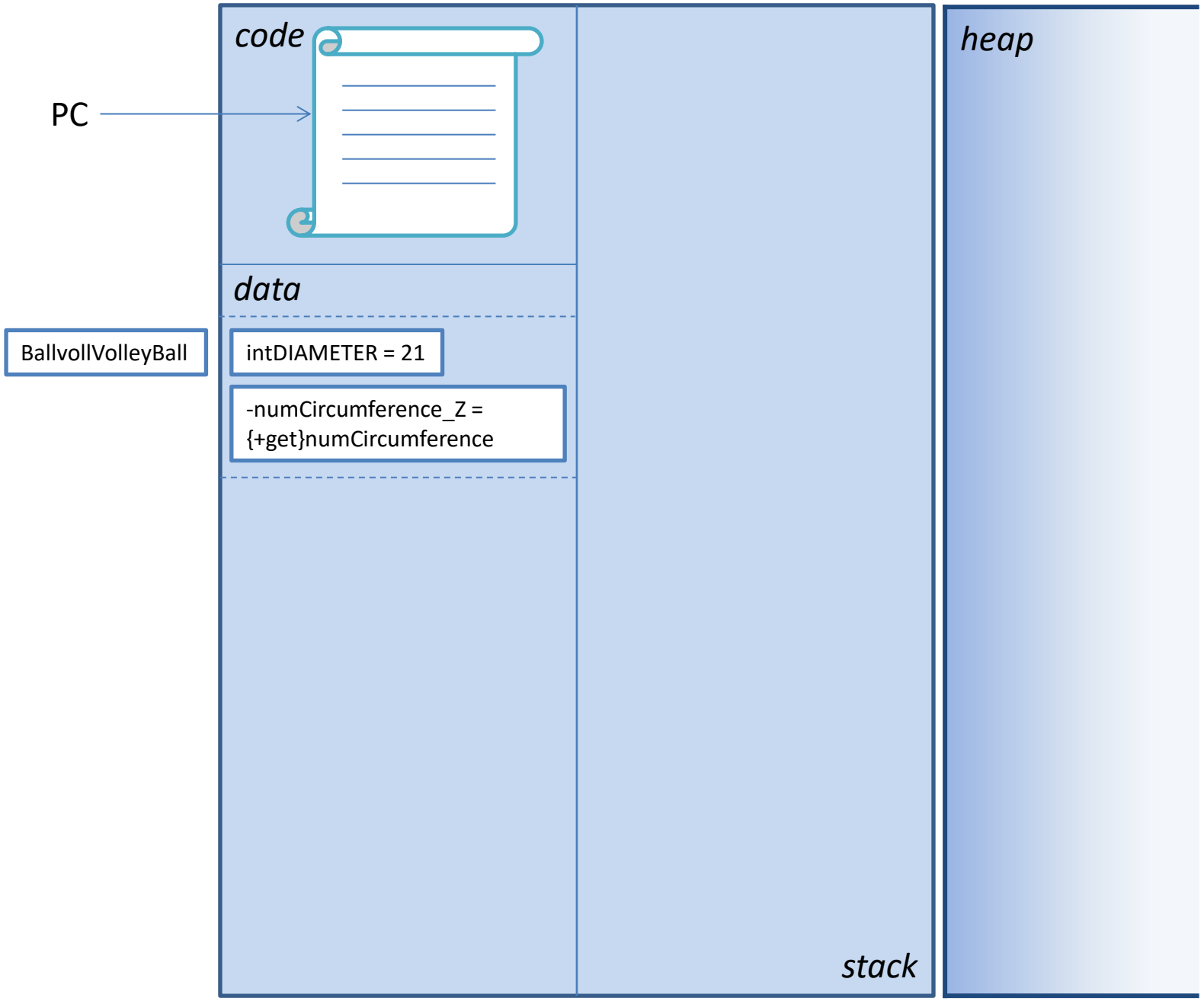

BallvollVolleyBall
+intDIAMETER = 21 : int
-numCircumference_Z : double
{+get}numCircumference : double
BallvollVolleyBall( )
-numCalculateCircumference() : double

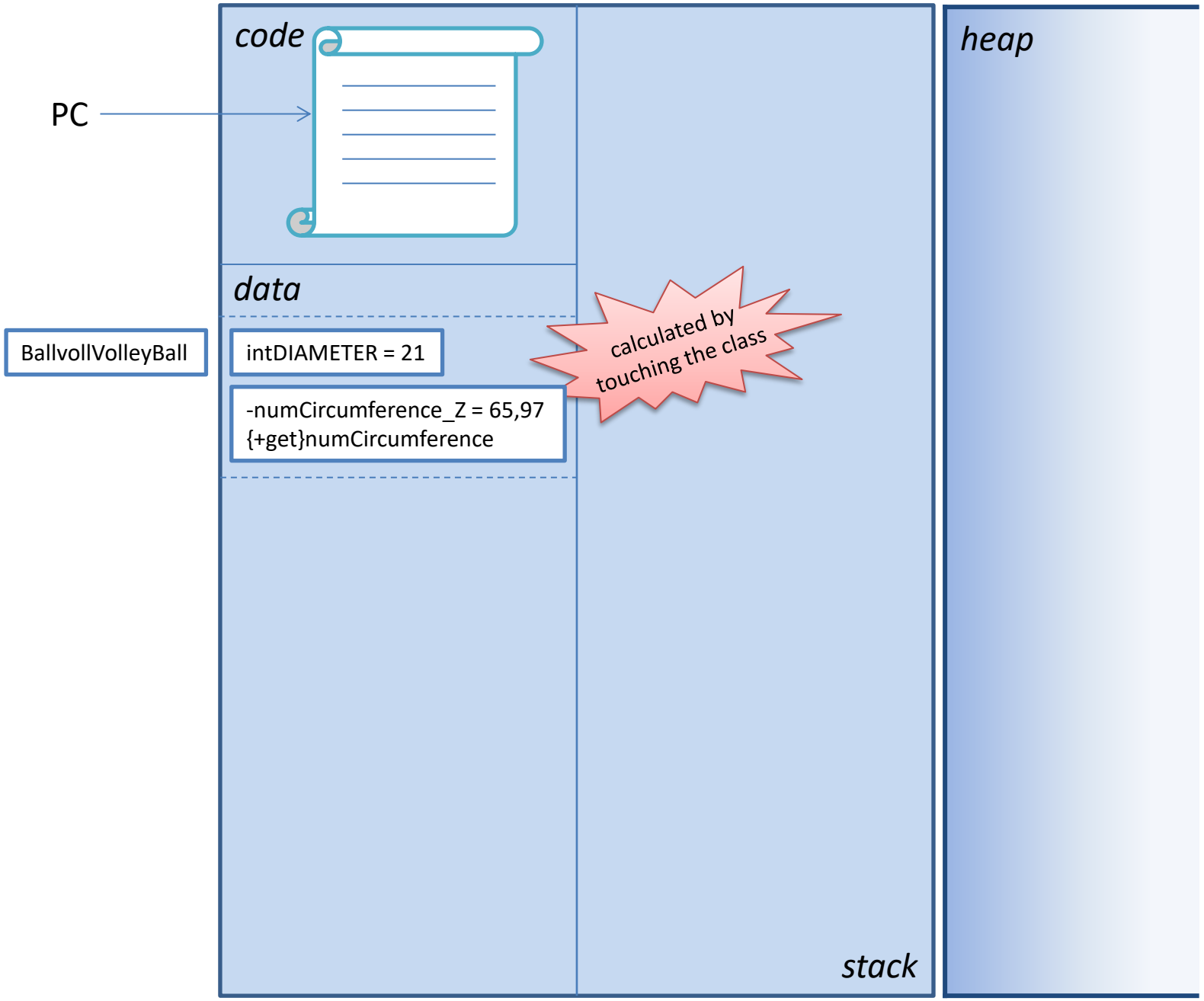
java

```
private static double xxx_Z;
public static double xxx() { return class.xxx_Z; }

static
{

}
```

C#

```
private static double xxx_Z;
public static double xxx { get { return class.xxx_Z; } }

static BallvollVolleyBall(
    )
{

}
```


code
PC
heap
data
BallvollVolleyBall
intDIAMETER = 21
-numCircumference_Z =
{+get}numCircumference
stack

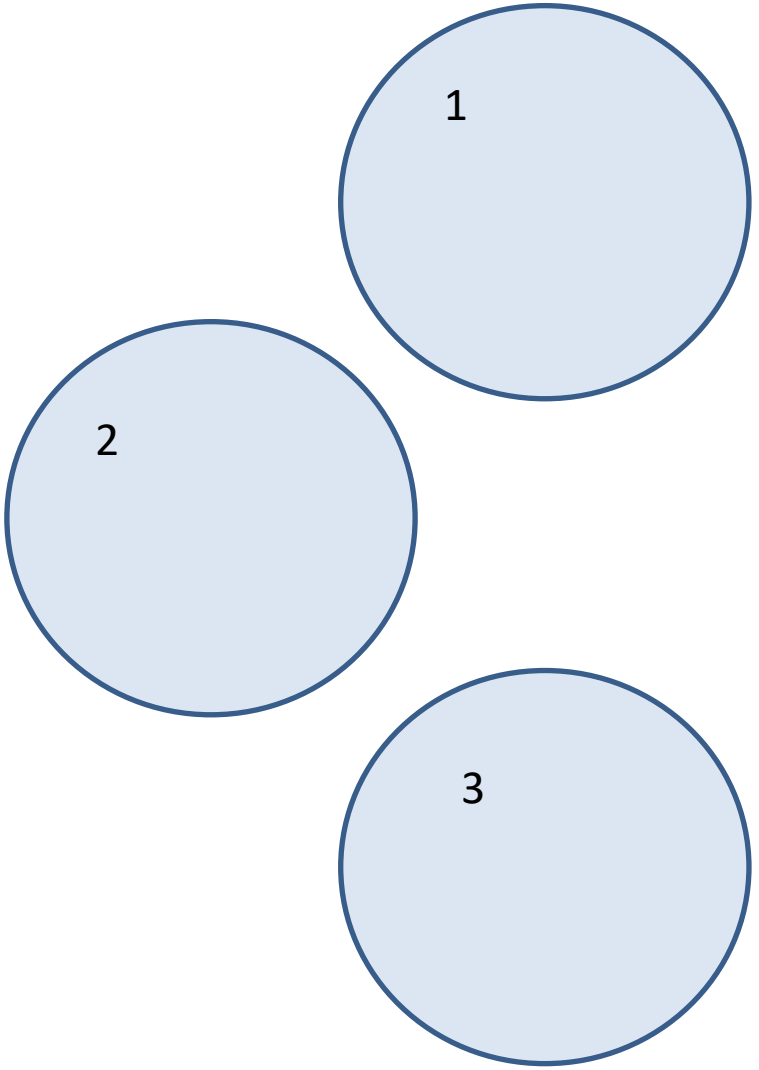

code
PC
data
heap
BallvollVolleyBall
intDIAMETER = 21
-numCircumference_Z = 65,97
{+get}numCircumference
calculated by touching the class
stack

Real world

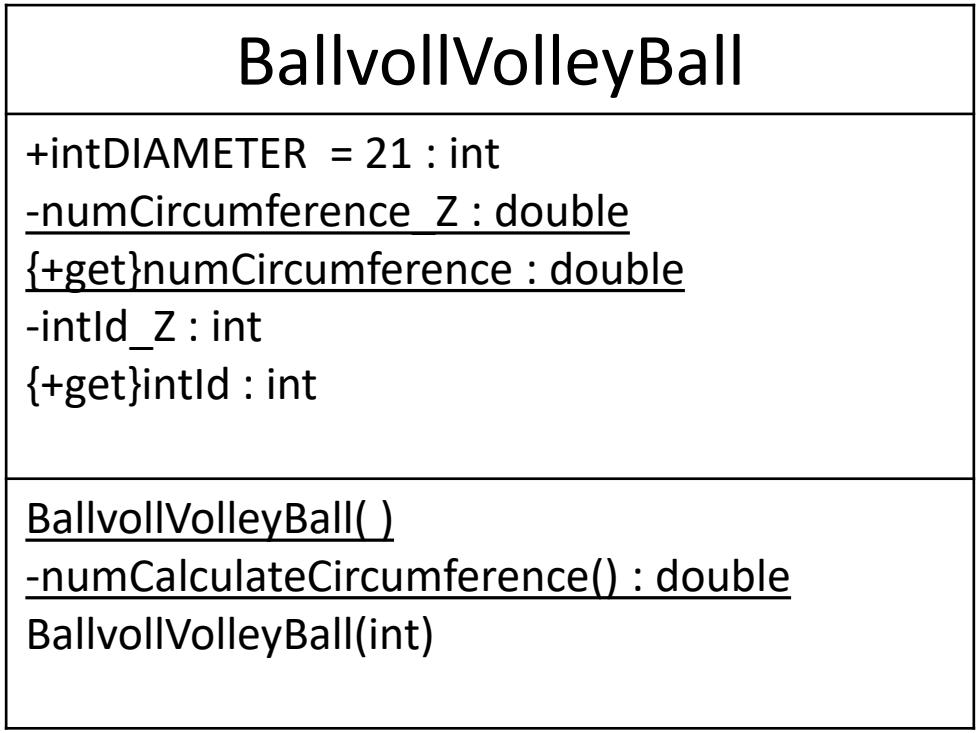
1

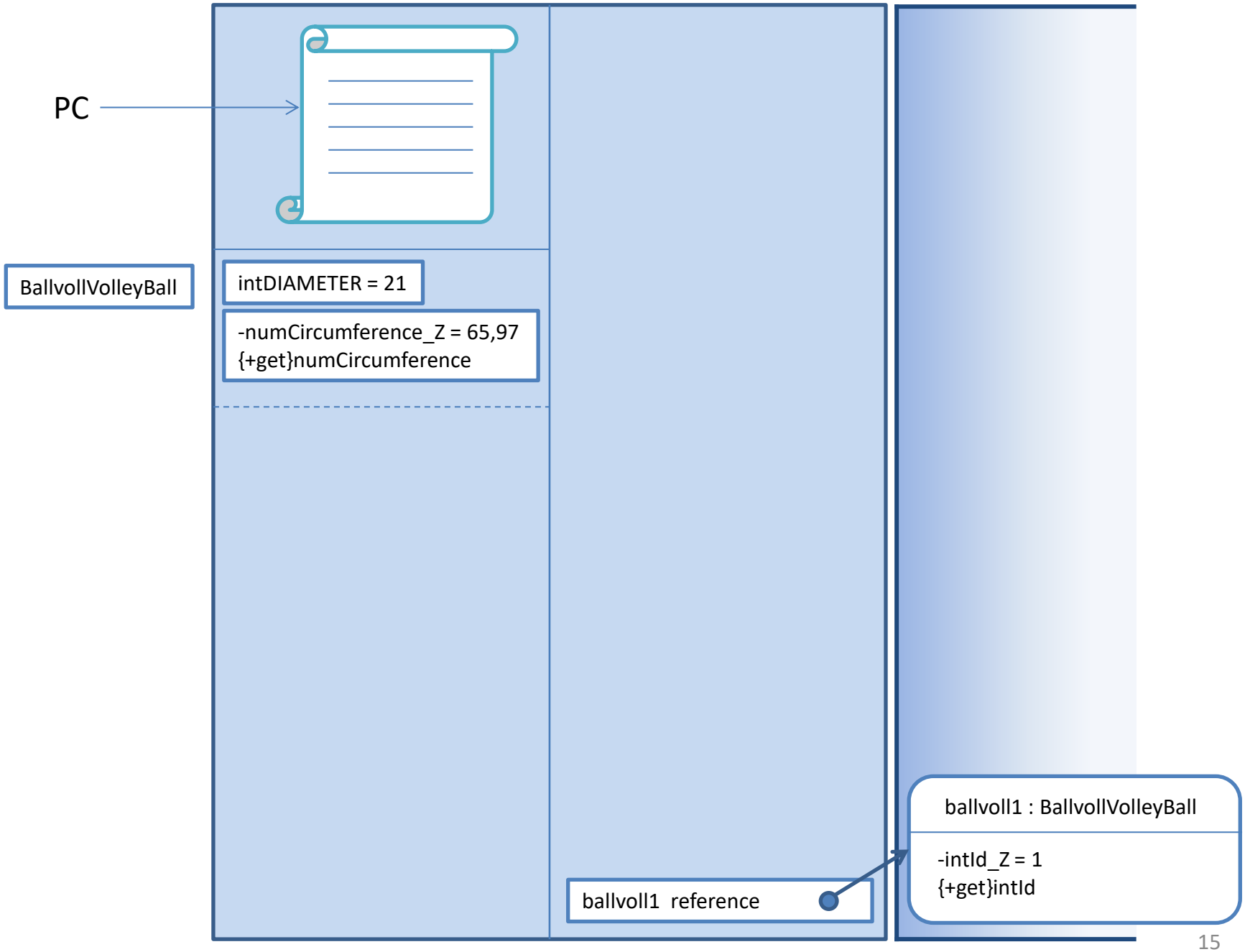
2

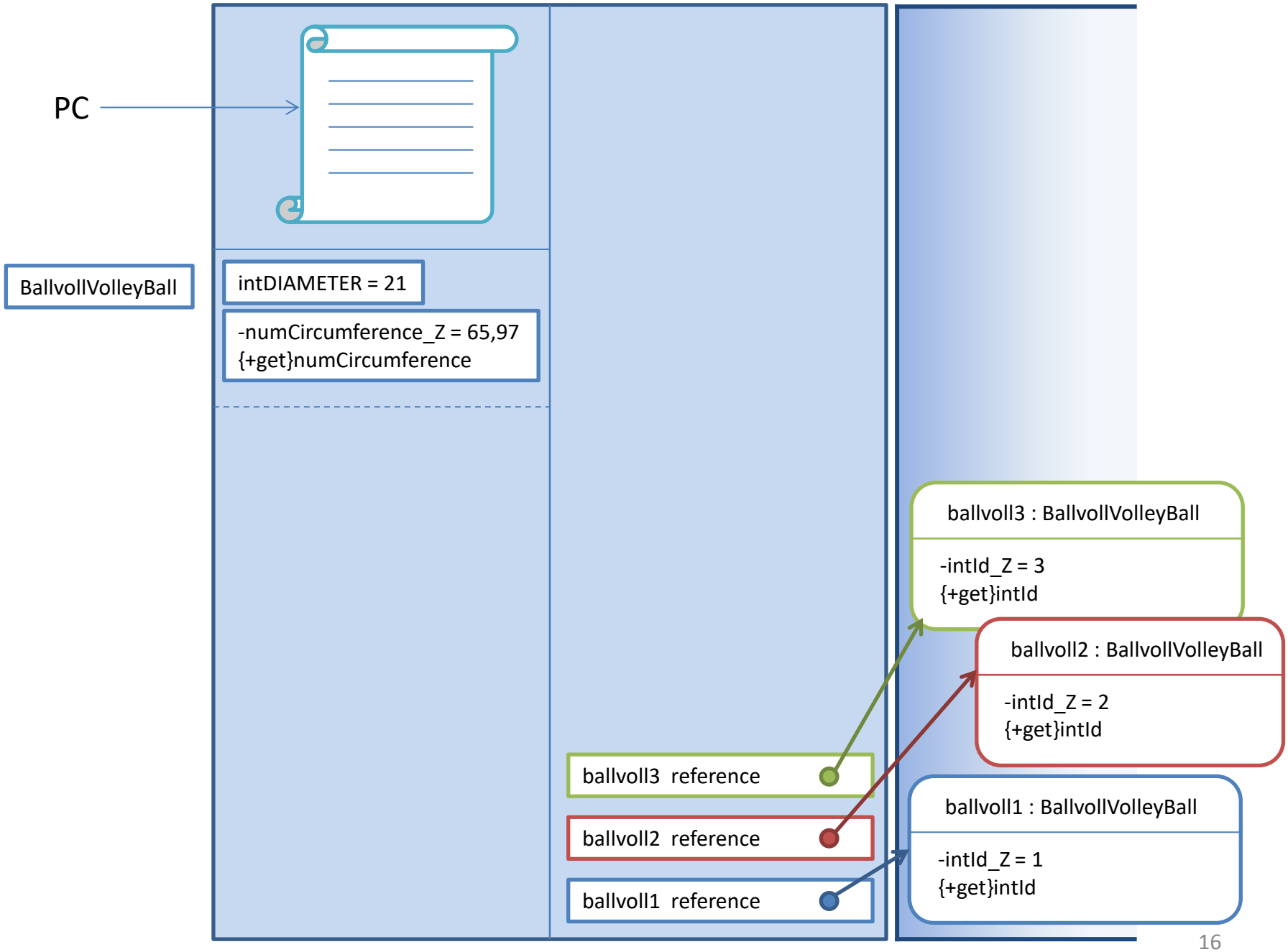
3



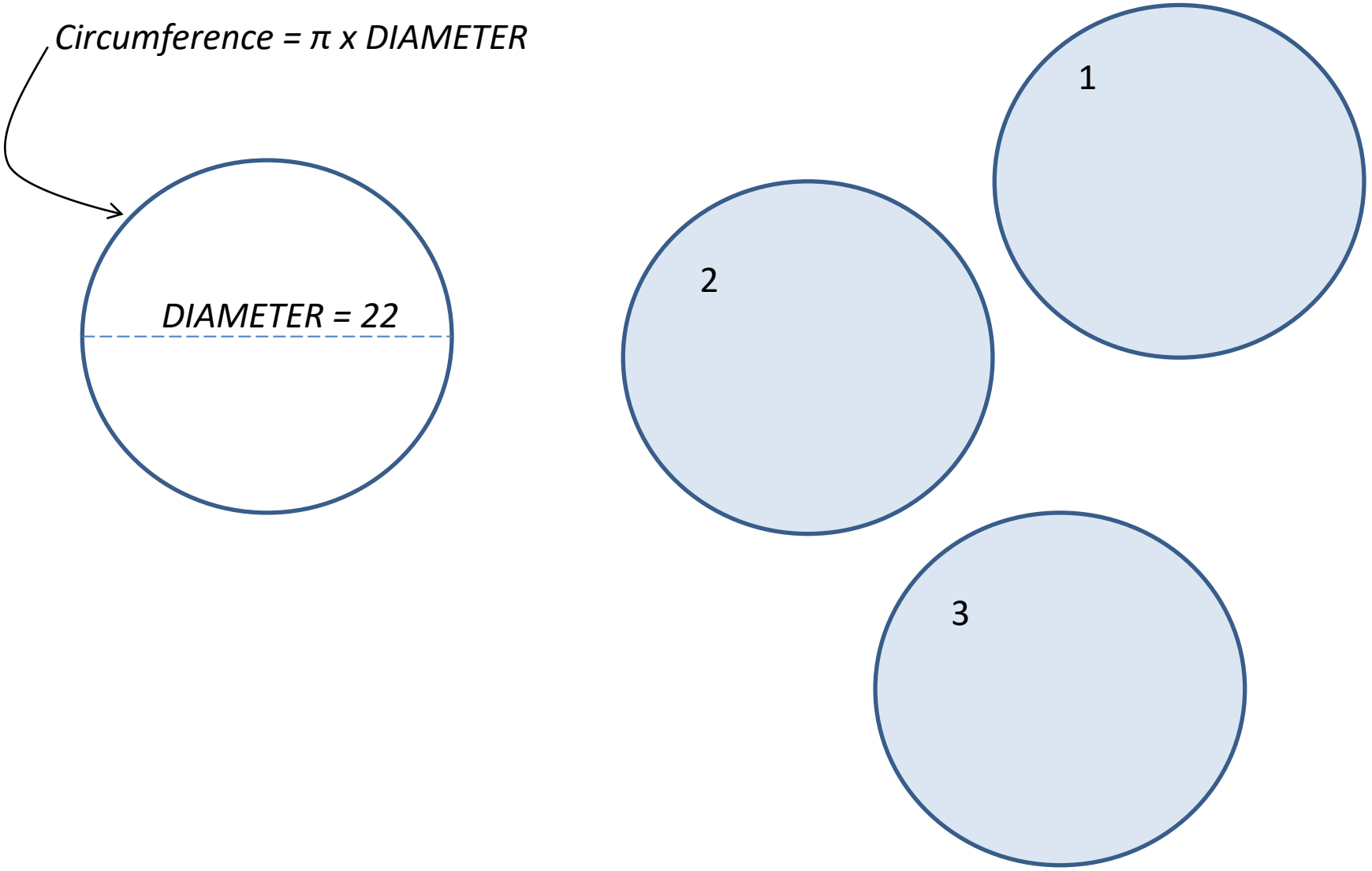
>> Challenge 4. Add an instance variable Id and a constructor


BallvollVolleyBall
+intDIAMETER = 21 : int
-numCircumference_Z : double
{+get}numCircumference : double
-intId_Z : int
{+get}intId : int
BallvollVolleyBall( )
-numCalculateCircumference() : double
BallvollVolleyBall(int)

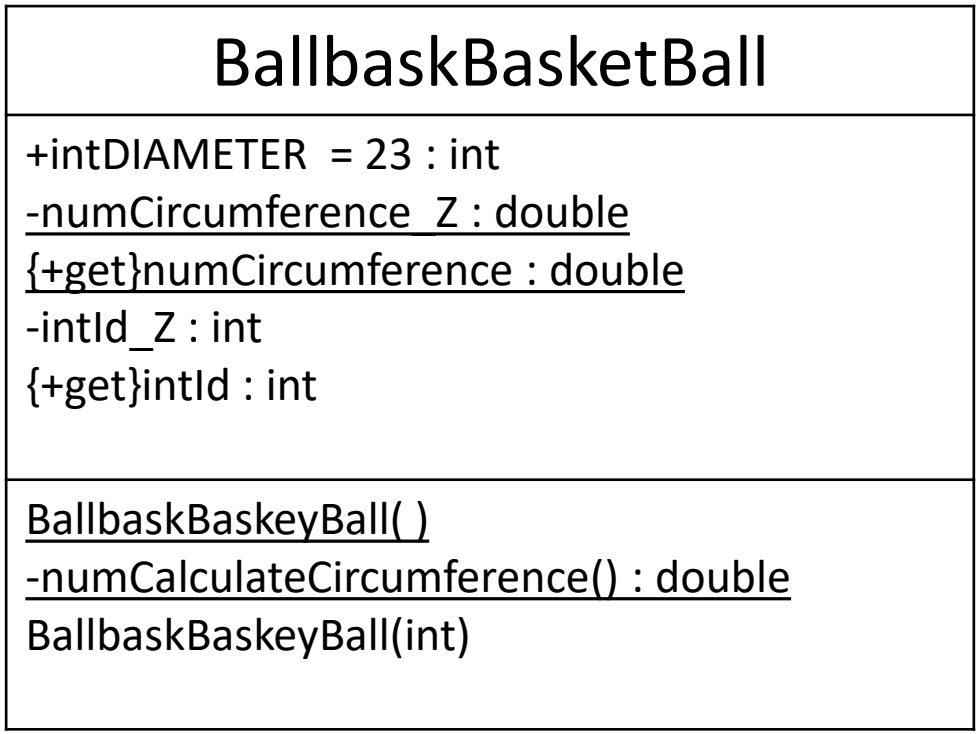

PC
BallvollVolleyBall
intDIAMETER = 21
-numCircumference_Z = 65,97
{+get}numCircumference
ballvoll1 reference
ballvoll1 : BallvollVolleyBall
-intId_Z = 1
{+get}intId


PC
BallvollVolleyBall
intDIAMETER = 21
-numCircumference_Z = 65,97
{+get}numCircumference
ballvoll3 : BallvollVolleyBall
-intId_Z = 3
{+get}intId
ballvoll2 : BallvollVolleyBall
-intId_Z = 2
{+get}intId
ballvoll1 : BallvollVolleyBall
-intId_Z = 1
{+get}intId
ballvoll3 reference
ballvoll2 reference
ballvoll1 reference

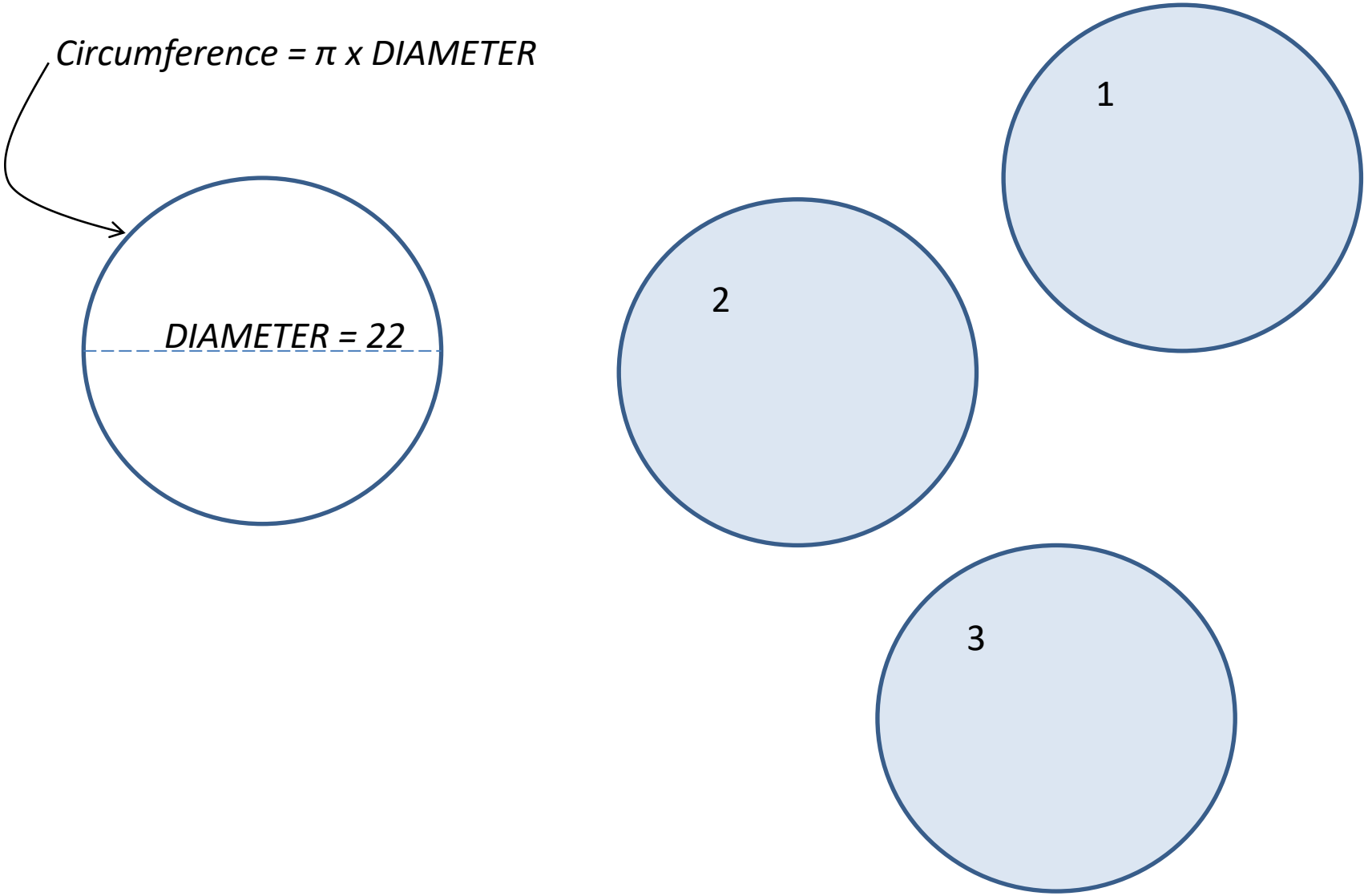
Real world



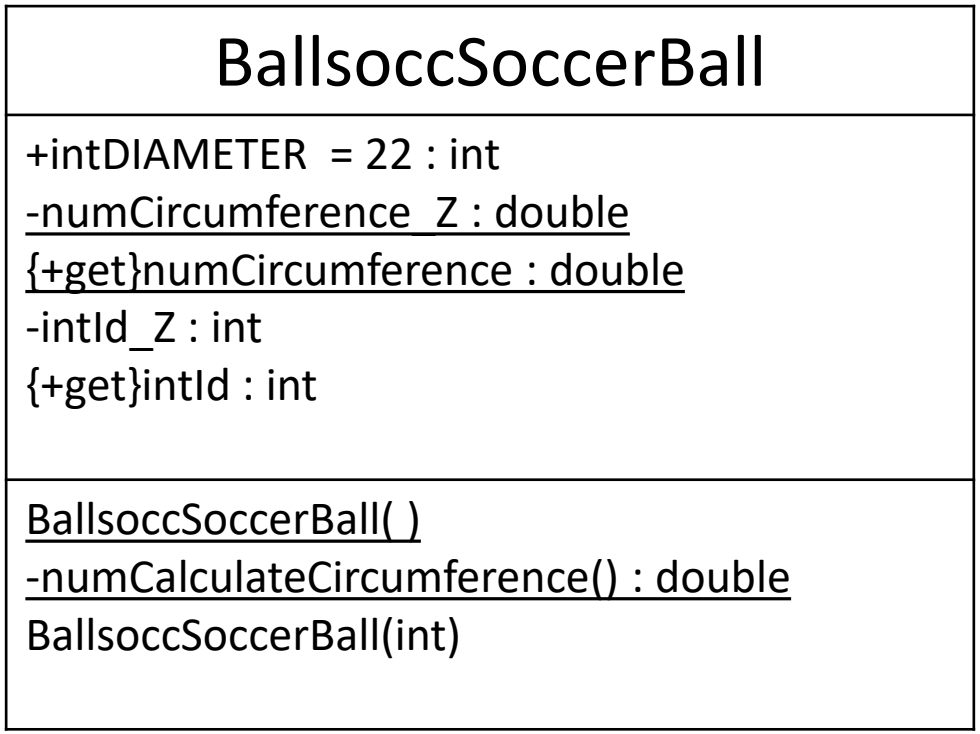
>> Challenge 5. Add a BasketBall class

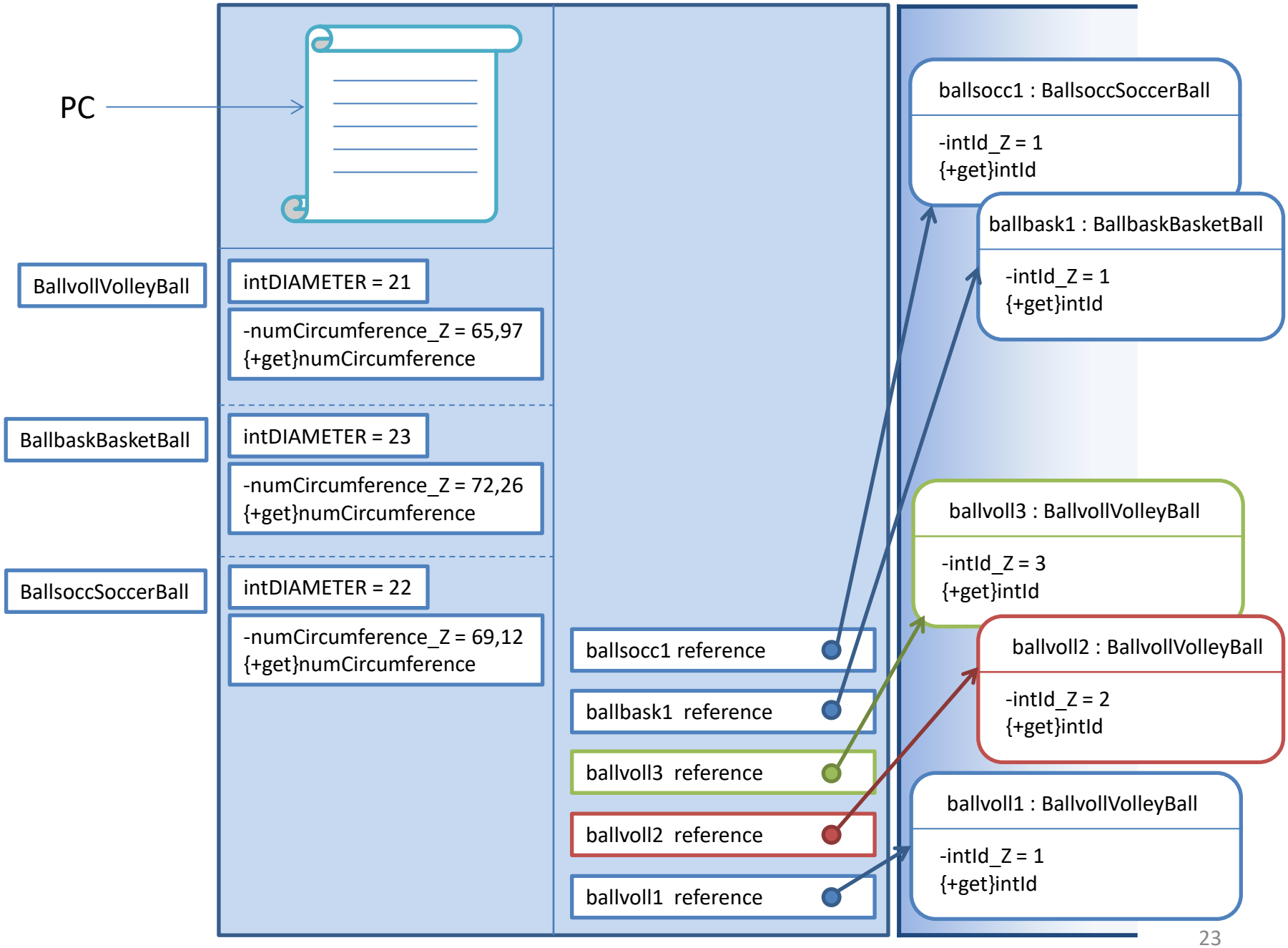

BallbaskBasketBall
+intDIAMETER = 23 : int
-numCircumference_Z : double
{+get}numCircumference : double
-intId_Z : int
{+get}intId : int
BallbaskBaskeyBall( )
-numCalculateCircumference() : double
BallbaskBaskeyBall(int)

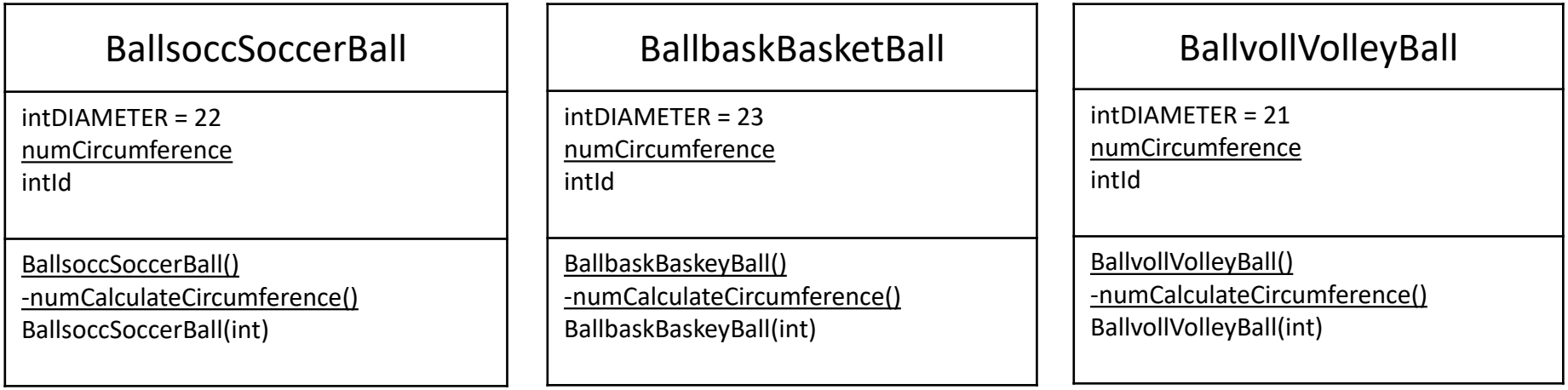
Real world

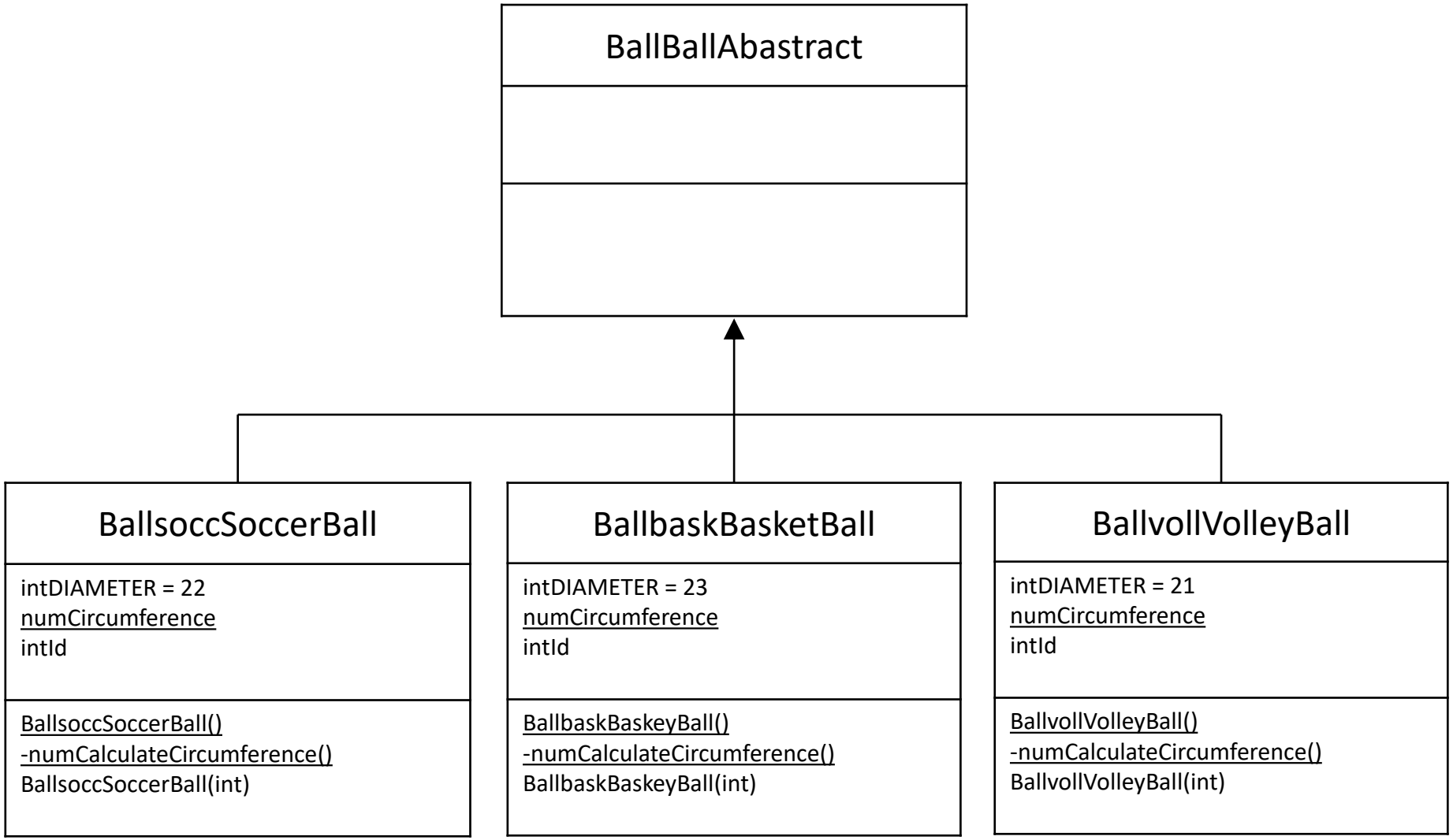


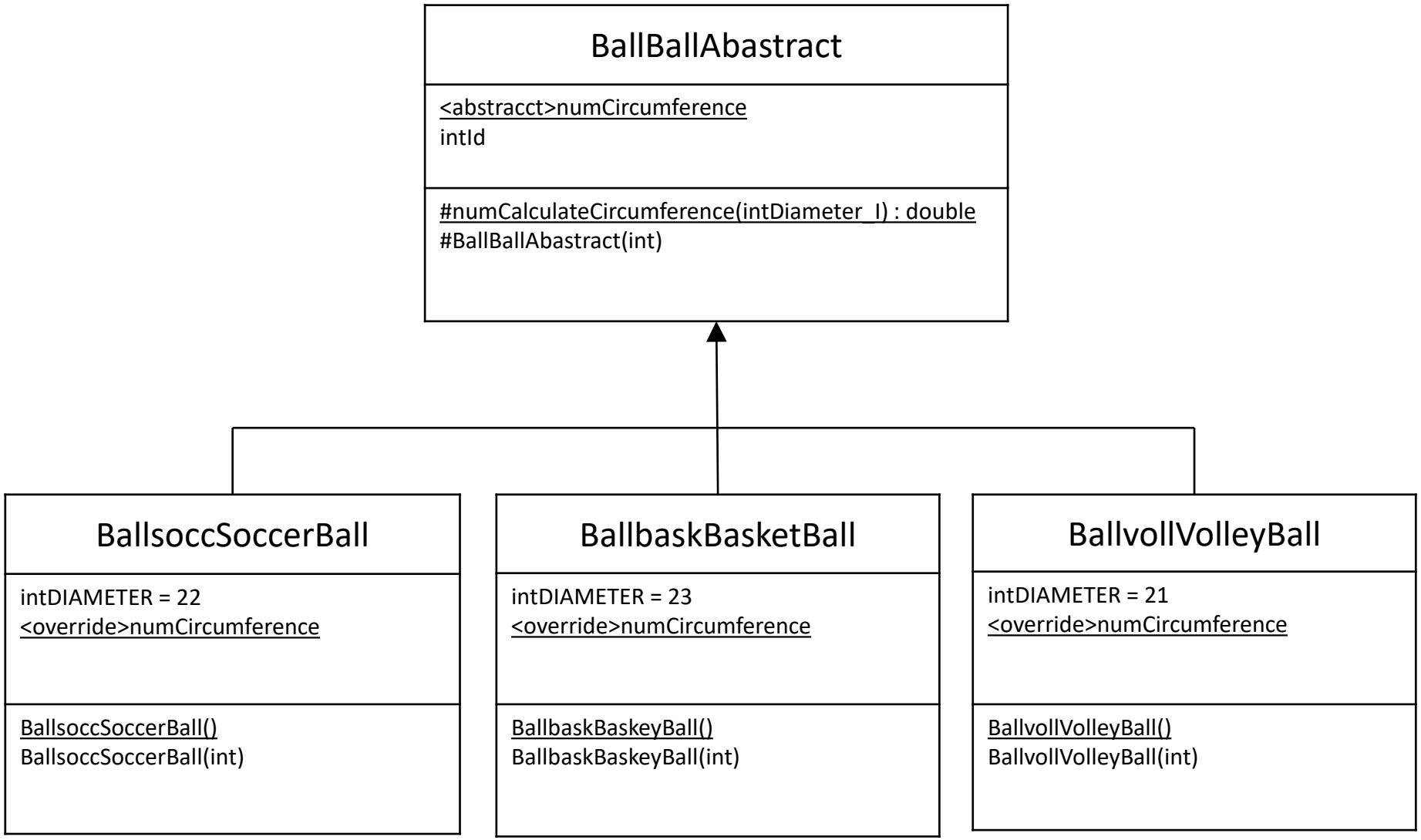
>> Challenge 6. Add a SoccerBall class

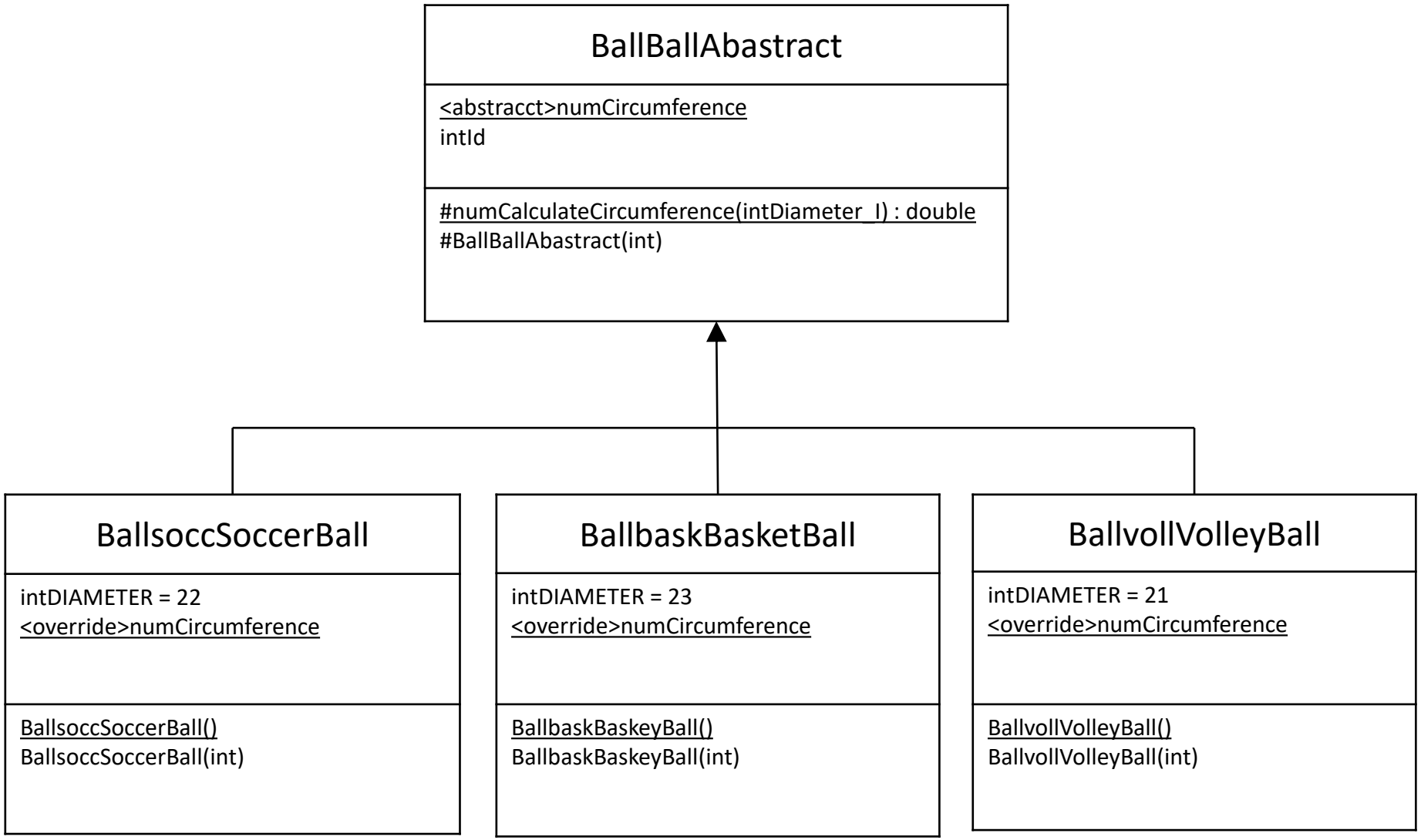

BallsoccSoccerBall
+intDIAMETER = 22 : int
-numCircumference_Z : double
{+get}numCircumference : double
-intId_Z : int
{+get}intId : int
BallsoccSoccerBall( )
-numCalculateCircumference() : double
BallsoccSoccerBall(int)


PC
BallvollVolleyBall
intDIAMETER = 21
-numCircumference_Z = 65,97
{+get}numCircumference
BallbaskBasketBall
intDIAMETER = 23
-numCircumference_Z = 72,26
{+get}numCircumference
BallsoccSoccerBall
intDIAMETER = 22
-numCircumference_Z = 69,12
{+get}numCircumference
ballsocc1 reference
ballbask1 reference
ballvoll3 reference
ballvoll2 reference
ballvoll1 reference
ballsocc1 : BallsoccSoccerBall
-intId_Z = 1
{+get}intId
ballbask1 : BallbaskBasketBall
-intId_Z = 1
{+get}intId
ballvoll3 : BallvollVolleyBall
-intId_Z = 3
{+get}intId
ballvoll2 : BallvollVolleyBall
-intId_Z = 2
{+get}intId
ballvoll1 : BallvollVolleyBall
-intId_Z = 1
{+get}intId


BallsoccSoccerBall
intDIAMETER = 22
numCircumference
intId
BallsoccSoccerBall()
-numCalculateCircumference()
BallsoccSoccerBall(int)
BallbaskBasketBall
intDIAMETER = 23
numCircumference
intId
BallbaskBaskeyBall()
-numCalculateCircumference()
BallbaskBaskeyBall(int)
BallvollVolleyBall
intDIAMETER = 21
numCircumference
intId
BallvollVolleyBall()
-numCalculateCircumference()
BallvollVolleyBall(int)


BallBallAbastract
BallsoccSoccerBall
intDIAMETER = 22
numCircumference
intId
BallsoccSoccerBall()
-numCalculateCircumference()
BallsoccSoccerBall(int)
BallbaskBasketBall
intDIAMETER = 23
numCircumference
intId
BallbaskBaskeyBall()
-numCalculateCircumference()
BallbaskBaskeyBall(int)
BallvollVolleyBall
intDIAMETER = 21
numCircumference
intId
BallvollVolleyBall()
-numCalculateCircumference()
BallvollVolleyBall(int)


BallBallAbastract
<abstracct>numCircumference
intId
#numCalculateCircumference(intDiameter_I) : double
#BallBallAbastract(int)
BallsoccSoccerBall
intDIAMETER = 22
<override>numCircumference
BallsoccSoccerBall()
BallsoccSoccerBall(int)
BallbaskBasketBall
intDIAMETER = 23
<override>numCircumference
BallbaskBaskeyBall()
BallbaskBaskeyBall(int)
BallvollVolleyBall
intDIAMETER = 21
<override>numCircumference
BallvollVolleyBall()
BallvollVolleyBall(int)



>> Challenge 7. Add the BallBallAbstrac class

To create balls…

new BallvollVolleyBall(1)
new BallbaskBasketBall(1)
new BallsoccSoccerBall(1)

I want a method that allows me to create objects of any kind of ball: VolleyBall, BasketBall or SoccerBall.

Can I use the constructor method of Ball?

```
BallBallAbstract Ball                    lBallAbstract(1);
```

No, abstract classes cannot be instantiated

I want a method that allows me to create objects of any kind of ball: VolleyBall, BasketBall or SoccerBall.

I should have a parameter that inicates what type of object I want to create.

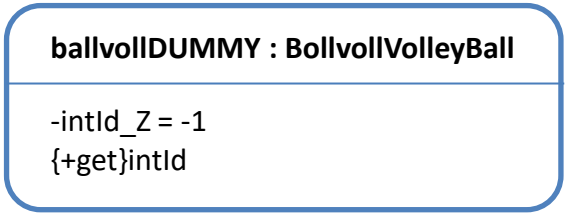
```
public static BallBallAbstract newball(
    int intId_I,
    BallBallAbstract ballDUMMY_I
    )
```

What about a dummy object?

What is a dummy?

It is an object with values without meaning
Use a parameter-less constructor

```
//-----------------------------------------------------------------
public BallvollVolleyBall(
        )
            : base(-1)
        {

        }
```



Let´s create a dummy

```
BallBallAbstract ballvollDUMMY = new BallvollVolleyBall();

System.Console.Write(String.Format(
"The ballvollDUMMY has a circumference of {0:N} \n",
ballvollDUMMY.numCircumference));
```

A dummy object has access to the instance methods

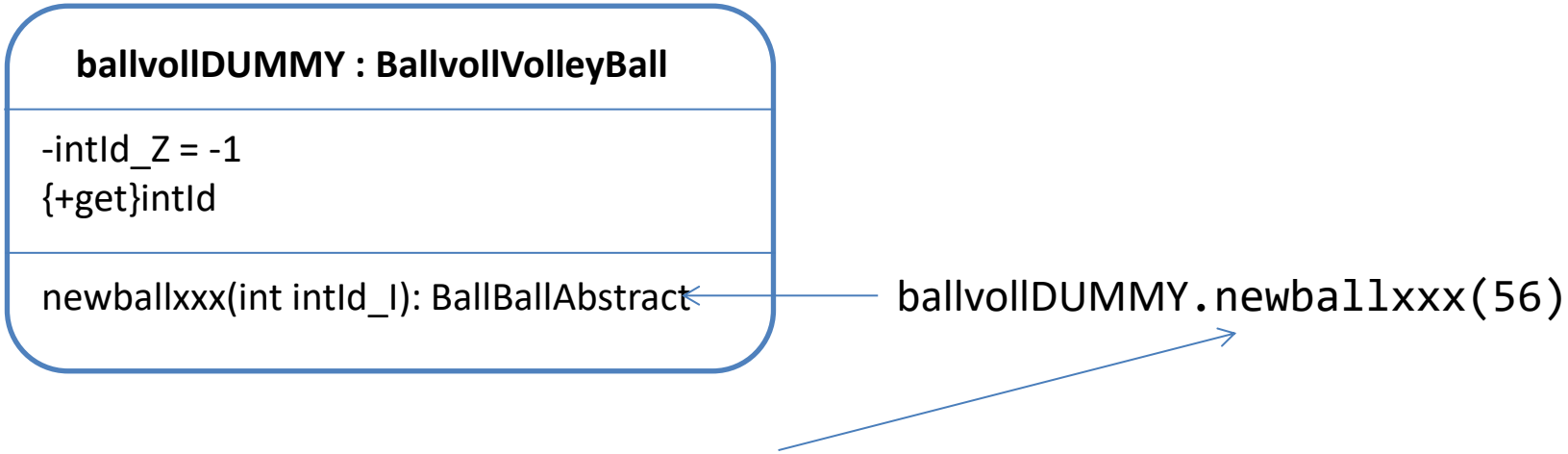
>> Challenge 8. Create dummy object for the classes

I want a method that allows me to create objects of any kind of ball: VolleyBall, BasketBall or SoccerBall.

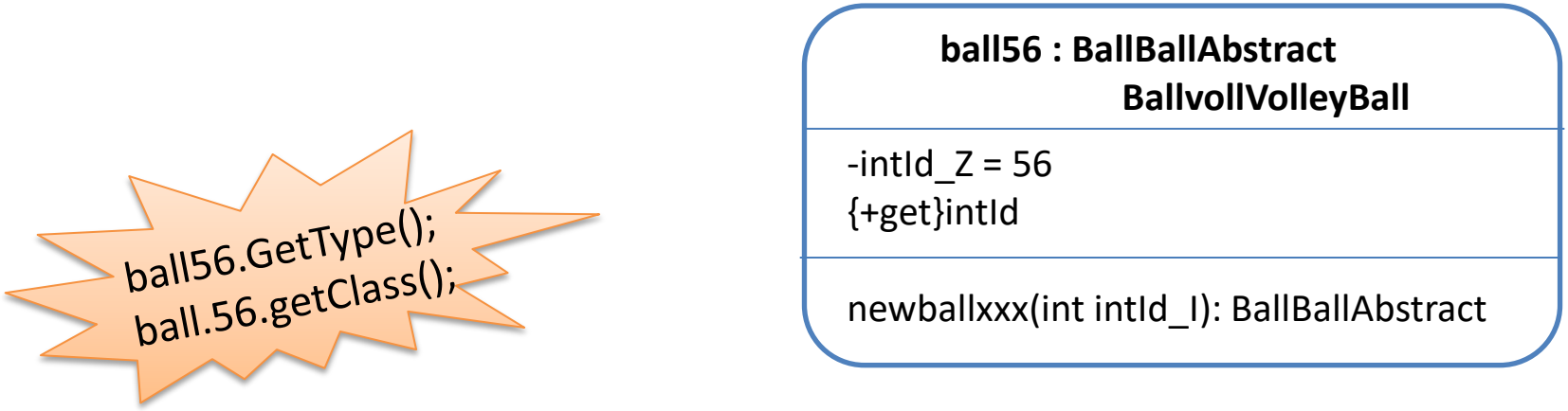
```
public static BallBallAbstract newball(
    int intId_I,
    BallBallAbstract ballDUMMY
    )
{
    BallBallAbstract Ball =  ballDUMMY.newballxxx(intId_I);
    return Ball;
}

public abstract BallBallAbstract newballxxx(int intId_I);
```

>> Challenge 9. Create an object factory for ball objects

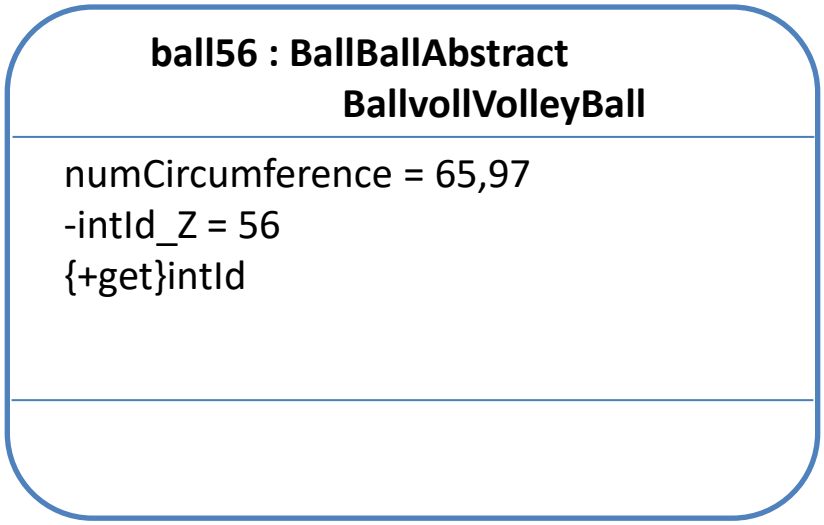
BallBallAbstract ballvollDUMMY = new BallvollVolleyBall();

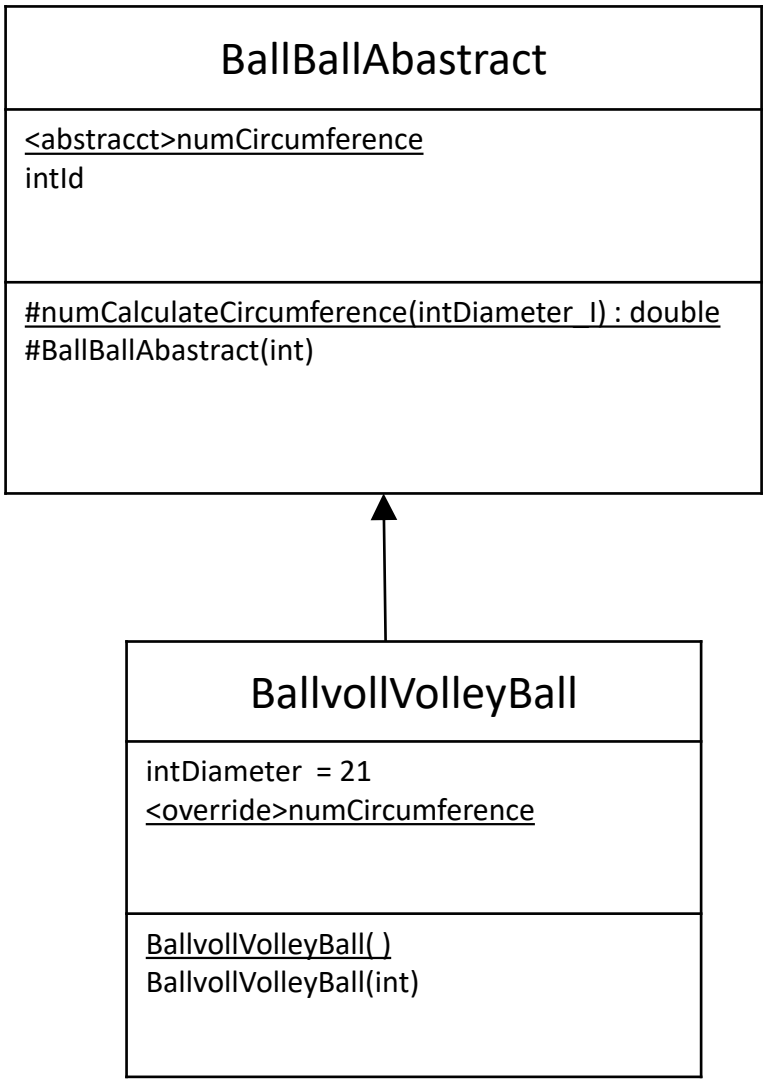


BallBallAbstract ball56 = BallBallAbstract.newball(56, ballvollDUMMY);



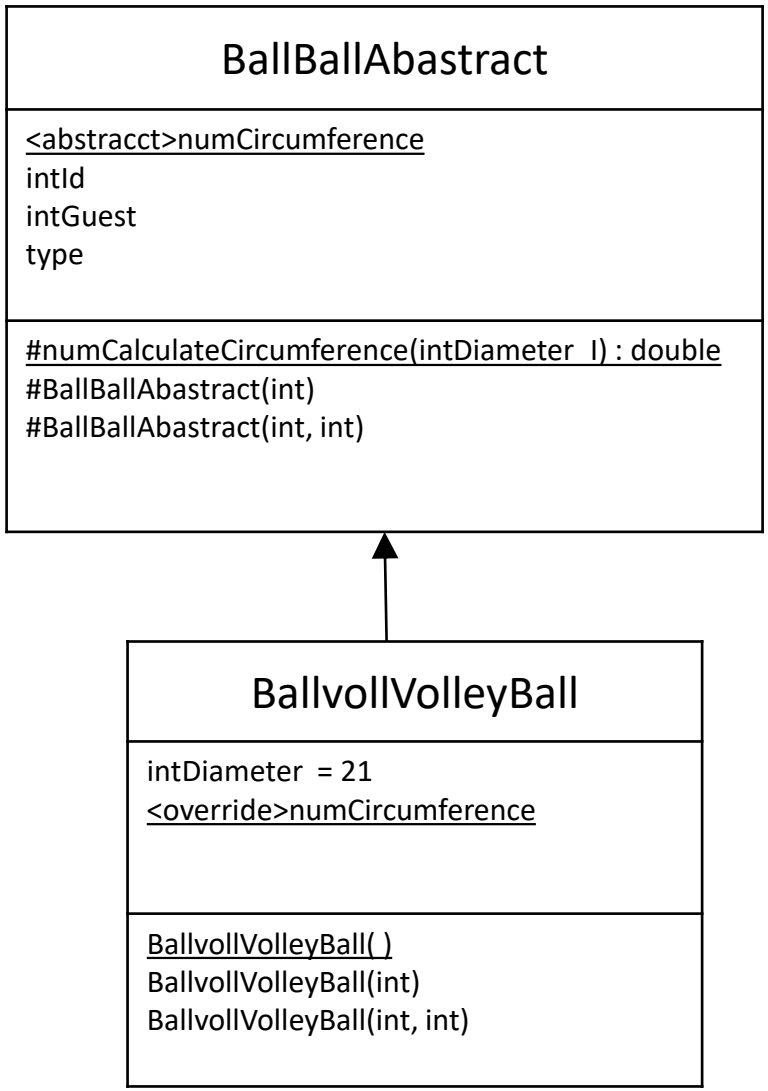
Real world


ball56 : BallBallAbstract
BallvollVolleyBall
numCircumference = 65,97
-intId_Z = 56
{+get}intId


BallBallAbastract
<abstracct>numCircumference
intId
#numCalculateCircumference(intDiameter_I) : double
#BallBallAbastract(int)
BallvollVolleyBall
intDiameter = 21
<override>numCircumference
BallvollVolleyBall( )
BallvollVolleyBall(int)

ball56 : BallBallAbstract
BallvollVolleyBall

numCircumference = 65,97
-intId_Z = 56
{+get}intId
intGuest
type



I need to know to which guest I lent the ball…

I would like to know the type of the ball…

>> Challenge 10. Add the guest and the type variables to the Ball class