```scala
package polynomial

import tinyscalautils.lang.InterruptibleConstructor

sealed abstract class HornerPoly extends InterruptibleConstructor:
   def eval(x: Double): Double
   def + (that: HornerPoly): HornerPoly
   def * (that: HornerPoly): HornerPoly
   def * (n: Int): HornerPoly
   def degree: Int
   def derivative: HornerPoly
   def toList: List[Int]
end HornerPoly

private def canonicalS(c: Int, q: HornerPoly): HornerPoly =
  if c == 0 then
    q match
      case C(0) => C(0)
      case _    => X(q)
  else  q match
    case C(0) => C(c)
    case _    => S(c, q)

private def canonicalX(q: HornerPoly): HornerPoly =
  q match
    case C(0) => C(0)
    case _    => X(q)

private final case class C(c: Int) extends HornerPoly:
   def eval(x: Double): Double          = c.toDouble

   def + (that: HornerPoly): HornerPoly = that match
     case C(n)      => C(c + n)
     case S(n, q)   => canonicalS(c + n, q)
     case X(q)      => canonicalS(c, q)

```

```
37    def * (that: HornerPoly): HornerPoly = that * c
38    def * (n: Int): HornerPoly           = if n == 0
   then C(0) else C(c * n)
39    def degree: Int                      = 0
40    def derivative: HornerPoly           = C(0)
41    def toList: List[Int]                = List(c)
42 end C
43
44 private final case class X(q: HornerPoly) extends
   HornerPoly:
45    def eval(x: Double): Double          = x * q.eval(x)
46
47    def + (that: HornerPoly): HornerPoly = that match
48      case C(n) =>
49        canonicalS(n, q)
50
51      case X(r) =>
52        canonicalX(q + r)
53
54      case S(n, r) =>
55        canonicalS(n, q + r)
56
57    def * (that: HornerPoly): HornerPoly =
58      that match
59        case C(n) =>
60          this * n
61
62        case _ =>
63          canonicalX(q * that)
64
65    def * (n: Int): HornerPoly           = if n == 0
   then C(0) else canonicalX(q * n)
66    def degree: Int                      = q.degree + 1
67    def derivative: HornerPoly           = q +
   canonicalX(q.derivative)
68    def toList: List[Int]                = 0 :: q.toList
69 end X
70
```

```scala
71 private final case class S(c: Int, q: HornerPoly)
   extends HornerPoly:
72   def eval(x: Double): Double          = c + x * q.
   eval(x)
73
74   def + (that: HornerPoly): HornerPoly = that match
75     case C(n) =>
76       canonicalS(c + n, q)
77
78     case S(n, r) =>
79       canonicalS(c + n, q + r)
80
81     case X(r) =>
82       val sum = q + r
83       canonicalS(c, sum)
84
85   def * (that: HornerPoly): HornerPoly = that match
86     case C(n) =>
87       this * n
88
89     case _ =>
90       val left  = canonicalX(q * that)
91       val right = that * c
92       left + right
93
94   def * (n: Int): HornerPoly           = if n == 0
   then C(0) else canonicalS(c * n, q * n)
95   def degree: Int                      = q.degree + 1
96   def derivative: HornerPoly           = q +
   canonicalX(q.derivative)
97   def toList: List[Int]                = c :: q.
   toList
98 end S
99
```