Project_5b: Scala Pet Store Simulation

Create a standalone Scala program that will successfully run if called using Scala.

Create an object hierarchy as follows (with attributes and methods()):

- Animal (name, sleep())
  - Feline (eat(), play(), makeNoise())
    - At least 3 named Cat instances
  - Canine (eat(), play(), makeNoise())
    - At least 3 named Dog instances
- Staff (arrive(), lunch(), leave())
  - Manager (openStore(),closeStore())
    - Named Manager instance
  - Clerk (feedAnimals(), playAnimals(), sellAnimals())
    - Named Clerk instance
- Clock (day, hour, announceHour())

You may implement Clock and Staff objects as you wish. You may create them in communicating threads, or you may just keep all objects in a single thread for their collaboration. (So multithreading is optional, communication between objects is required.) The Store will have a collection of three named Cat instances and three named Dog instances.

The Clock object will send announceHour events to the Staff as follows for any given day:

- 0800: Manager instance will arrive and openStore. The openStore method causes all animal instances to makeNoise.
- 0900: Clerk instance will arrive and feedAnimals. The feedAnimals method causes all animal instances to eat.
- 1100: Clerk instance will playAnimals. The playAnimals method causes all animal instances to play.
- 1200: Manager instance performs lunch method.
- 1300: Staff instance performs lunch method.
- 1600: Clerk instance will sellAnimals. The sellAnimals method checks each of the animals to see if they were sold today. There is a 20% chance of an animal being sold. If the animal is sold, announce the sale, and replace the sold animal with a new named instance of the animal.
- 1700: Clerk instance performs leave method.
- 1800: Manager instance performs closeStore and leaves. The closeStore method causes all Animal instances to sleep.

Every method, when executed by an object, should display an appropriate string, like:

- “1100: Clem the Clerk plays with the animals.”
- “1100: Cookie the Cat plays and chases their tail.”

Run New Format Clear Messages Worksheet Downloa

```scala
import scala.collection.mutable.ListBuffer
import scala.util.Random

// Abstract class for animals
abstract class Animal(var name: String) {
  def makeNoise(): Unit
  def eat(): Unit
  def play(): Unit
  def sleep(): Unit = println(s"$name sleeps.")
}

// Class for dogs
class Leo(name: String) extends Animal(name) {
  override def makeNoise(): Unit = println(s"$name says Ruff.")
  override def eat(): Unit = println(s"$name munches on dog food.")
  override def play(): Unit = println(s"$name fetches the ball.")
}

// Class for cats
class Luna(name: String) extends Animal(name) {
  override def makeNoise(): Unit = println(s"$name says Meow.")
  override def eat(): Unit = println(s"$name munches on cat food.")
  override def play(): Unit = println(s"$name chases a light.")
}

// Trait for staff members
trait Staff {
  val name: String
  def arrive(): Unit
  def lunch(): Unit
  def leave(): Unit
}

// Manager class
class Manager(val name: String) extends Staff {
  def arrive(): Unit = println(s"$name the Manager arrives.")
```

Run New Format Clear Messages Worksheet Download

```scala
// Manager class
class Manager(val name: String) extends Staff {
  def arrive(): Unit = println(s"$name the Manager arrives.")
  def openStore(animals: ListBuffer[Animal]): Unit = {
    println(s"$name opens the store, animals start making noise:")
    animals.foreach(_.makeNoise())
  }
  def lunch(): Unit = println(s"$name has lunch.")
  def closeStore(): Unit = println(s"$name closes the store.")
  def leave(): Unit = println(s"$name leaves the store.")
}

// Clerk class
class Clerk(val name: String, animals: ListBuffer[Animal]) extends Staff {
  def arrive(): Unit = println(s"$name the Clerk arrives.")
  def feedAnimals(): Unit = {
    println(s"$name feeds the animals:")
    animals.foreach(_.eat())
  }
  def playAnimals(): Unit = {
    println(s"$name plays with the animals:")
    animals.foreach(_.play())
  }
  def sellAnimals(): Unit = {
    println(s"$name attempts to sell an animal.")
    if (Random.nextDouble() < 0.2) {
      val index = Random.nextInt(animals.length)
      val animal = animals.remove(index)
      println(s"An animal was sold: ${animal.name}")
      val newAnimal = if (animal.isInstanceOf[Luna]) {
        new Luna("Kitten" + Random.nextInt(100))
      } else {
        new Leo("Pup" + Random.nextInt(100))
      }
      animals += newAnimal
    } else {
```

Run New Format Clear Messages Worksheet Download Embed

```
      animals += newAnimal
    } else {
      println(s"$name was unsuccessful at selling an animal.")
    }
  }
  def lunch(): Unit = println(s"$name has lunch.")
  def leave(): Unit = println(s"$name leaves the store.")
}

// Main object
object PetStoreSimulation {
  def main(args: Array[String]): Unit = {
    // List of animals in the pet store
    val animals = ListBuffer[Animal](
      new Luna("Rea"), new Luna("Milo"), new Luna("Lily"),
      new Leo("Max"), new Leo("Luke"), new Leo("Abby")
    )
    // Create manager and clerk
    val manager = new Manager("James")
    val clerk = new Clerk("Charles", animals)

    // Daily schedule
    val dailySchedule = List("0800", "0900", "1100", "1200", "1300", "1600", "1700", "1800")

    // Simulate for 3 days
    for (day <- 1 to 3) {
      dailySchedule.foreach { time =>
        println(s"--- Day $day at $time ---")
        time match {
          case "0800" =>
            manager.arrive()
            manager.openStore(animals)
          case "0900" =>
            clerk.arrive()
            clerk.feedAnimals()
          case "1100" =>
```

Run New Format Clear Messages Worksheet Download

```
    // Simulate for 3 days
    for (day <- 1 to 3) {
      dailySchedule.foreach { time =>
        println(s"--- Day $day at $time ---")
        time match {
          case "0800" =>
            manager.arrive()
            manager.openStore(animals)
          case "0900" =>
            clerk.arrive()
            clerk.feedAnimals()
          case "1100" =>
            clerk.playAnimals()
          case "1200" =>
            manager.lunch()
          case "1300" =>
            clerk.lunch()
          case "1600" =>
            clerk.sellAnimals()
            println("Current Animal Inventory:")
            animals.foreach(animal => print(s" - ${animal.name}"))
            println()
          case "1700" =>
            clerk.leave()
          case "1800" =>
            manager.closeStore()
            animals.foreach(_.sleep())
            manager.leave()
          case _ => // no action
        }
      }
    }
  }
}

```

```
--- Day 1 at 0800 ---
James the Manager arrives.
James opens the store, animals start making noise:
Rea says Meow.
Milo says Meow.
Lily says Meow.
Max says Ruff.
Luke says Ruff.
Abby says Ruff.
```

```
--- Day 1 at 0900 ---
Charles the Clerk arrives.
Charles feeds the animals:
Rea munches on cat food.
Milo munches on cat food.
Lily munches on cat food.
Max munches on dog food.
Luke munches on dog food.
Abby munches on dog food.
--- Day 1 at 1100 ---
```

```
--- Day 1 at 1100 ---
Charles plays with the animals:
Rea chases a light.
Milo chases a light.
Lily chases a light.
Max fetches the ball.
Luke fetches the ball.
Abby fetches the ball.
```

```
--- Day 1 at 1200 ---
James has lunch.
--- Day 1 at 1300 ---
Charles has lunch.
--- Day 1 at 1600 ---
Charles attempts to sell an animal.
Charles was unsuccessful at selling an animal.
Current Animal Inventory:
 - Rea - Milo - Lily - Max - Luke - Abby
--- Day 1 at 1700 ---
Charles leaves the store.
```

```
--- Day 1 at 1800 ---
James closes the store.
Rea sleeps.
Milo sleeps.
Lily sleeps.
Max sleeps.
Luke sleeps.
Abby sleeps.
James leaves the store.
```

```
--- Day 2 at 0800 ---
James the Manager arrives.
James opens the store, animals start making noise:
Rea says Meow.
Milo says Meow.
Lily says Meow.
Max says Ruff.
Luke says Ruff.
Abby says Ruff.
--- Day 2 at 0900 ---
Charles the Clerk arrives.
Charles feeds the animals:
Rea munches on cat food.
Milo munches on cat food.
Lily munches on cat food.
Max munches on dog food.
```

```
Charles the Clerk arrives.
Charles feeds the animals:
Rea munches on cat food.
Milo munches on cat food.
Lily munches on cat food.
Max munches on dog food.
Luke munches on dog food.
Abby munches on dog food.
--- Day 2 at 1100 ---
Charles plays with the animals:
Rea chases a light.
Milo chases a light.
Lily chases a light.
Max fetches the ball.
Luke fetches the ball.
Abby fetches the ball.
```

```
--- Day 2 at 1200 ---
James has lunch.
--- Day 2 at 1300 ---
Charles has lunch.
--- Day 2 at 1600 ---
Charles attempts to sell an animal.
Charles was unsuccessful at selling an animal.
Current Animal Inventory:
 - Rea - Milo - Lily - Max - Luke - Abby
--- Day 2 at 1700 ---
Charles leaves the store.
--- Day 2 at 1800 ---
James closes the store.
Rea sleeps.
Milo sleeps.
Lily sleeps.
```

```
Rea sleeps.
Milo sleeps.
Lily sleeps.
Max sleeps.
Luke sleeps.
Abby sleeps.
James leaves the store.
--- Day 3 at 0800 ---
James the Manager arrives.
James opens the store, animals start making noise:
Rea says Meow.
Milo says Meow.
Lily says Meow.
Max says Ruff.
Luke says Ruff.
Abby says Ruff.
```

```
--- Day 3 at 0900 ---
Charles the Clerk arrives.
Charles feeds the animals:
Rea munches on cat food.
Milo munches on cat food.
Lily munches on cat food.
Max munches on dog food.
Luke munches on dog food.
Abby munches on dog food.
--- Day 3 at 1100 ---
Charles plays with the animals:
Rea chases a light.
Milo chases a light.
Lily chases a light.
Max fetches the ball.
Luke fetches the ball.
```

```
Charles plays with the animals:
Rea chases a light.
Milo chases a light.
Lily chases a light.
Max fetches the ball.
Luke fetches the ball.
Abby fetches the ball.
--- Day 3 at 1200 ---
James has lunch.
--- Day 3 at 1300 ---
Charles has lunch.
--- Day 3 at 1600 ---
Charles attempts to sell an animal.
Charles was unsuccessful at selling an animal.
Current Animal Inventory:
 - Rea - Milo - Lily - Max - Luke - Abby
```

```
Charles attempts to sell an animal.
Charles was unsuccessful at selling an animal.
Current Animal Inventory:
 - Rea - Milo - Lily - Max - Luke - Abby
--- Day 3 at 1700 ---
Charles leaves the store.
--- Day 3 at 1800 ---
James closes the store.
Rea sleeps.
Milo sleeps.
Lily sleeps.
Max sleeps.
Luke sleeps.
Abby sleeps.
James leaves the store.
```

Simulate the store running for three days, capturing output to a text file. Comment all code appropriately, including identifying header information, internal comments as warranted, and capturing URLs for citations of helpful or copied sources.

(30 points) Scala code (petStore.scala) for a correctly operating petStore class, including appropriate comments and cited support.

(20 points) Example of correct execution as captured results from a console display, including using scala to run the .scala source file. Output should go into a text file called petStoreResults.txt. Could also do a screen capture to an image file.

(10 points) A README file in Markdown including (1) Project title, (2) your name, (3) three elements you liked about using Scala for the project, (4) three elements of Scala you did not like (at least not as much), and (5) the most important reference you used including a URL (or citation).