Задание 1 из 9

Начало формы

Рассмотрим следующее определение:

**trait** **Product** {

**def** uuid = java.util.UUID.randomUUID.toString

**def** name: String

**def** description: String

**def** price: BigDecimal

}

Какие методы необходимо реализовать конкретному классу, расширяющему Product?

annotation

title

name

uuid

author

description

price

Среди следующих вариантов отметьте корректные подклассы Product.

Вариант 1:

**class** **Book**(val title: String,

val annotation: String,

val author: String,

val isbn: String,

val price: BigDecimal)

**extends** Product {

**def** name = title

**def** description = annotation

}

Вариант 2:

**class** **Book**(val name: String,

val description: String,

val author: String,

val isbn: String,

val price: BigDecimal)

**extends** Product

Вариант 3:

**class** **Book**

**extends** Product {

**var** name: String

**var** description: String

**var** author: String

**var** isbn: String

**var** price: BigDecimal

}

Вариант 4:

**abstract** **class** **Book**

**extends** Product {

**def** title: String

**def** author: String

**def** annotation: String

**def** isbn: String

**def** name = title

**def** description = annotation

}

Конец формы

Задание 3 из 9

Начало формы

Рассмотрим следующие определения.

*// Время*

**class** **Time**(val hours: Int,

val minutes: Int,

val seconds: Int)

*// Часы*

**trait** **Clock** {

*// Текущее время*

**def** getTime: Time

}

*// Наручные часы*

**class** **Watch** **extends** **Clock** {

**def** getTime = {

**val** secs = System.currentTimeMillis / 1000l

**val** seconds = secs % 60

**val** minutes = (secs / 60) % 60

**val** hours = (secs / 3600) % 24

**new** Time(hours.toInt, minutes.toInt, seconds.toInt)

}

}

Отметьте верные утверждения.

Класс Time расширяет Clock.

Класс Clock расширяет Time.

Класс Clock расширяет Watch.

Класс Watch расширяет Clock.

Отметьте корректные определения, расширяющие Clock.

Вариант 1:

**object** **StoppedWatch** **extends** **Watch** {

**override** **def** getTime = **new** Time(15, 20, 1)

}

Вариант 2:

**trait** **AlarmWatch** **extends** **Watch** {

**def** alarmTime: Time

}

Вариант 3:

**class** **AlarmWatch** **extends** **Watch** {

**def** alarmTime: Time

}

Вариант 4:

**abstract** **class** **AlarmClock**(val alarmTime: Time)

**extends** Clock

Вариант 5:

**trait** **AlarmWatch** **extends** **Watch** {

**def** alarmTime: Time

}

Конец формы

# Задание 4 из 9

Начало формы

1. Рассмотрим следующие определения.
2. *// Напиток*
3. **trait** **Beverage** {
4. *// Цвет*
5. **def** color: String
6. *// Концентрация кофеина*
7. **def** caffeinePerLitre: Double
8. }
9. *// Чай*
10. **trait** **Tea** **extends** **Beverage** {
11. **def** caffeinePerLitre = 0.16

}

*// Кофе*

**object** **Coffee** **extends** **Beverage** {

**def** color = "black"

**def** caffeinePerLitre = 0.20

}

*// Черный чай*

**object** **BlackTea** **extends** **Tea** {

**def** color = "brown"

}

*// Зеленый чай*

**object** **GreenTea** **extends** **Tea** {

**def** color = "light-yellow"

}

Отметьте корректные выражения.

val b: Beverage = Tea

val b: BlackTea = Tea

val b: Tea = BlackTea

val b: Coffee = Coffee

val b: BlackTea = GreenTea

val b: Beverage = Coffee

2. Рассмотрим следующее определение.

*// Кружка для напитков*

**class** **Mug**[+**B** <: **Beverage**](val beverage: B,

val volume: Double) {

**def** caffeine = volume \* beverage.caffeinePerLitre

}

Отметьте корректные выражения.

val m: Mug[Coffee] = new Mug(Coffee, 0.250)

val m: Mug[GreenTea] = new Mug(BlackTea, 0.250)

val m: Mug[BlackTea] = new Mug(Tea, 0.250)

val m: Mug[Coffee] = new Mug(Beverage, 0.250)

val m: Mug[Beverage] = new Mug(Coffee, 0.250)

val m: Mug[Coffee] = new Mug(BlackTea, 0.250)

val m: Mug[Tea] = new Mug(BlackTea, 0.250)

Конец формы

# Задание 5 из 9

Начало формы

Рассмотрим следующие определения.

**sealed** **trait** **Resource** {

**def** goldEquiv: Int

}

**class** **Gold** **extends** **Resource** {

**def** goldEquiv = 1

}

**trait** **MaterialResource** **extends** **Resource** {

**def** goldEquiv = 50

}

**class** **Ore** **extends** **MaterialResource**

**class** **Wood** **extends** **MaterialResource**

**trait** **PreciousResource** **extends** **Resource** {

**def** goldEquiv = 200

}

**class** **Gem** **extends** **PreciousResource**

**class** **Crystal** **extends** **PreciousResource**

**class** **Sulfur** **extends** **PreciousResource**

**class** **Mercury** **extends** **PreciousResource**

Отметьте верные утверждения.

Ближайший общий надтип классов Wood и Mercury — PreciousResource.

Ближайший общий надтип классов Gem и Sulfur — PreciousResource.

Ближайший общий надтип классов Gold, Wood и Gem — Resource.

Перечисленные классы и интерфейсы не могут быть расширены в других файлах.

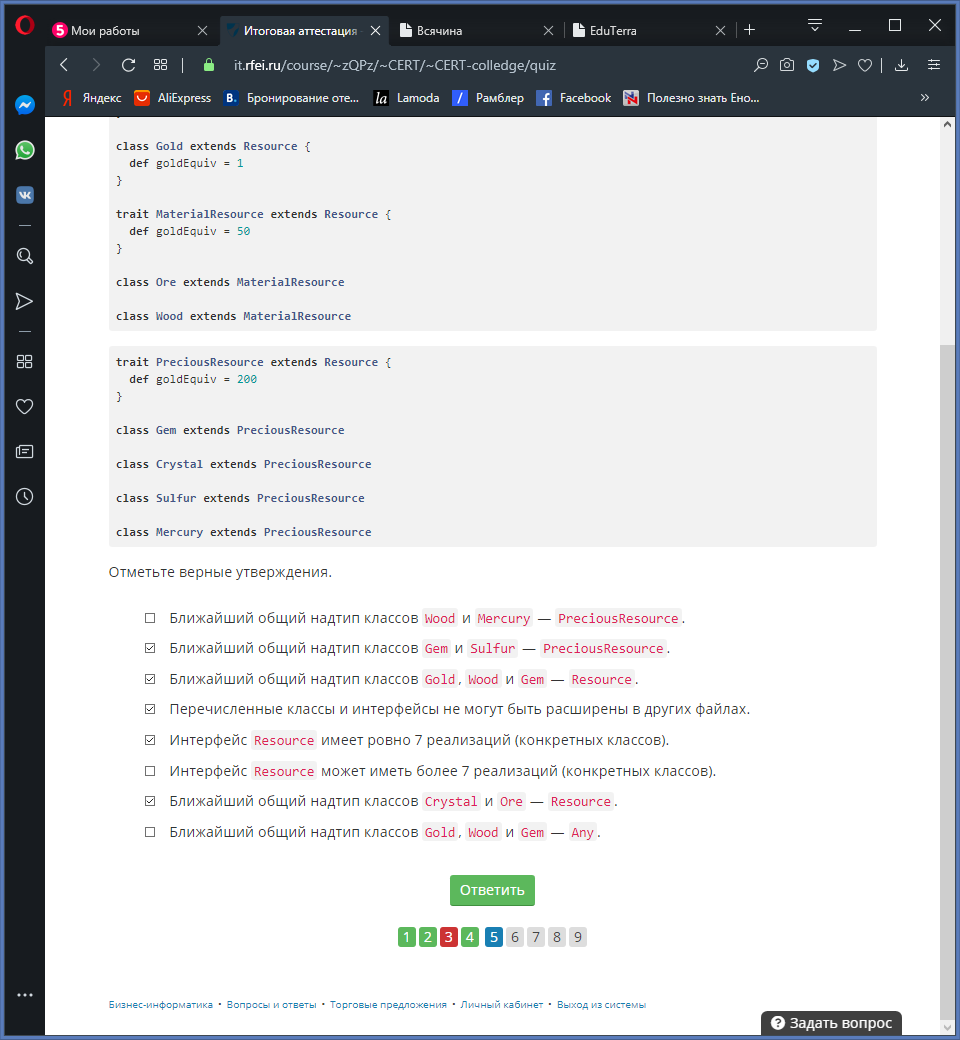
Интерфейс Resource имеет ровно 7 реализаций (конкретных классов).

Интерфейс Resource может иметь более 7 реализаций (конкретных классов).

Ближайший общий надтип классов Crystal и Ore — Resource.

Ближайший общий надтип классов Gold, Wood и Gem — Any.

Конец формы



# Задание 6 из 9

Начало формы

Рассмотрим следующие Scala-файлы.

Файл blog.scala:

**package** com.myblog

**package** model

**class** **Blog**(var title: String) {

**protected** **val** \_posts: ListBuffer[Post]

**def** feed = \_posts.toSeq.sortBy(\_.date).reverse

**def** publish(post: Post) {

\_posts += post

post.date = **new** Date

}

}

Файл post.scala:

**package** com.myblog

**package** model

**class** **Post**(var text: String,

var date: Date)

Отметьте верные утверждения.

Поля класса Post неизменяемые.

Классы пакета com.myblog видны в обоих файлах.

Класс Post расширяет класс Blog.

Классы пакета com.myblog.model видны в обоих файлах.

Класс Post неизменяемый.

Поля класса Blog неизменяемые.

Класс Blog расширяет класс Post.

Класс Blog неизменяемый.

# Задание 8 из 9

Начало формы

Рассмотрим следующие Scala-файлы.

Файл customer.scala:

**package** com.myshop

**package** model

**import** collection.mutable.ListBuffer

**class** **Customer**(val name: String,

val address: String) {

**protected** **val** \_orders = **new** ListBuffer[Order]

**def** orders = \_orders.toSeq

**def** add(order: Order) {

\_orders += order

}

}

Файл order.scala:

**package** com.myshop.model

**class** **Order**(val customer: Customer,

val status: String,

val date: Date)

Отметьте верные утверждения.

Класс Order неизменяемый.

Класс Order расширяет класс Customer.

Классы пакета com.myshop.model видны в обоих файлах.

Класс Customer неизменяемый.

Поля класса Order неизменяемые.

Класс Customer расширяет класс Order.

Классы пакета com.myshop видны в обоих файлах.

Поля класса Customer неизменяемые.

Конец формы

# Задание 9 из 9

Начало формы

Рассмотрим следующие определения.

*// Коробка*

**trait** **Box** {

**def** width: Double

**def** height: Double

**def** depth: Double

**final** **def** volume = width \* height \* depth

}

*// Спичка*

**class** **Match**

*// Спичечный коробок*

**class** **MatchBox** **extends** **Box** {

**def** width = 0.035

**def** height = 0.012

**def** depth = 0.05

**protected** **var** \_count = 50

**def** getMatch(): Option[Match] =

**if** (\_count > 0) {

\_count -= 1

Some(**new** Match)

} **else** None

}

Отметьте верные утверждения.

MatchBox расширяет Box.

Интерфейс Box является параметризованным.

Box расширяет MatchBox.

Класс MatchBox неизменяемый.

Метод volume можно переопределить в MatchBox.

Методы класса MatchBox не имеют сторонних эффектов.

Все методы интерфейса Box абстрактны.

Метод volume не может быть переопределен в подтипах Box.

Конец формы

## **Задание 1 00:07:14**

Рассмотрим следующее определение:

**trait** **Product** {

**def** uuid = java.util.UUID.randomUUID.toString

**def** name: String

**def** description: String

**def** price: BigDecimal

}

Какие методы необходимо реализовать конкретному классу, расширяющему Product?

author

annotation

title

uuid

description

price

name

Среди следующих вариантов отметьте корректные подклассы Product.

Вариант 1:

**class** **Book**(val title: String,

val annotation: String,

val author: String,

val isbn: String,

val price: BigDecimal)

**extends** Product {

**def** name = title

**def** description = annotation

}

Вариант 2:

**class** **Book**(val name: String,

val description: String,

val author: String,

val isbn: String,

val price: BigDecimal)

**extends** Product

Вариант 3:

**class** **Book**

**extends** Product {

**var** name: String

**var** description: String

**var** author: String

**var** isbn: String

**var** price: BigDecimal

}

Вариант 4:

**abstract** **class** **Book**

**extends** Product {

**def** title: String

**def** author: String

**def** annotation: String

**def** isbn: String

**def** name = title

**def** description = annotation

}

## **Задание 2 00:16:55**

Рассмотрим следующие определения.

**trait** **Person** {

**def** name: String

}

**class** **Teacher**(val name: String)

**extends** Person

**class** **Student**(val name: String)

**extends** Person

**class** **Mark**(val student: Student,

val teacher: Teacher,

val subject: String,

val markId: String,

val date: Date)

Отметьте верные утверждения.

Класс Student реализует Person.

Класс Mark расширяет Student.

Класс Student неизменяемый.

Класс Mark расширяет Teacher.

Класс Teacher расширяет Mark.

Класс Mark неизменяемый.

Класс Teacher реализует Person.

Класс Teacher неизменяемый.

Класс Student расширяет Mark.

## **Задание 3 00:06:58**

Рассмотрим следующие определения.

*// Время*

**class** **Time**(val hours: Int,

val minutes: Int,

val seconds: Int)

*// Часы*

**trait** **Clock** {

*// Текущее время*

**def** getTime: Time

}

*// Наручные часы*

**class** **Watch** **extends** **Clock** {

**def** getTime = {

**val** secs = System.currentTimeMillis / 1000l

**val** seconds = secs % 60

**val** minutes = (secs / 60) % 60

**val** hours = (secs / 3600) % 24

**new** Time(hours.toInt, minutes.toInt, seconds.toInt)

}

}

Отметьте верные утверждения.

Класс Time расширяет Clock.

Класс Clock расширяет Time.

Класс Watch расширяет Clock.

Класс Clock расширяет Watch.

Отметьте корректные определения, расширяющие Clock.

Вариант 1:

**object** **StoppedWatch** **extends** **Watch** {

**override** **def** getTime = **new** Time(15, 20, 1)

}

Вариант 2:

**trait** **AlarmWatch** **extends** **Watch** {

**def** alarmTime: Time

}

Вариант 3:

**class** **AlarmWatch** **extends** **Watch** {

**def** alarmTime: Time

}

Вариант 4:

**abstract** **class** **AlarmClock**(val alarmTime: Time)

**extends** Clock

Вариант 5:

**trait** **AlarmWatch** **extends** **Watch** {

**def** alarmTime: Time

}

## **Задание 4 00:05:56**

1. Рассмотрим следующие определения.
2. *// Напиток*
3. **trait** **Beverage** {
4. *// Цвет*
5. **def** color: String
6. *// Концентрация кофеина*
7. **def** caffeinePerLitre: Double
8. }
9. *// Чай*
10. **trait** **Tea** **extends** **Beverage** {
11. **def** caffeinePerLitre = 0.16

}

*// Кофе*

**object** **Coffee** **extends** **Beverage** {

**def** color = "black"

**def** caffeinePerLitre = 0.20

}

*// Черный чай*

**object** **BlackTea** **extends** **Tea** {

**def** color = "brown"

}

*// Зеленый чай*

**object** **GreenTea** **extends** **Tea** {

**def** color = "light-yellow"

}

Отметьте корректные выражения.

val b: BlackTea = Tea

val b: Beverage = Tea

val b: Tea = BlackTea

val b: BlackTea = GreenTea

val b: Beverage = Coffee

val b: Coffee = Coffee

2. Рассмотрим следующее определение.

*// Кружка для напитков*

**class** **Mug**[+**B** <: **Beverage**](val beverage: B,

val volume: Double) {

**def** caffeine = volume \* beverage.caffeinePerLitre

}

Отметьте корректные выражения.

val m: Mug[Beverage] = new Mug(Coffee, 0.250)

val m: Mug[GreenTea] = new Mug(BlackTea, 0.250)

val m: Mug[Coffee] = new Mug(Coffee, 0.250)

val m: Mug[Coffee] = new Mug(BlackTea, 0.250)

val m: Mug[BlackTea] = new Mug(Tea, 0.250)

val m: Mug[Tea] = new Mug(BlackTea, 0.250)

val m: Mug[Coffee] = new Mug(Beverage, 0.250)

## **Задание 5 00:08:46**

Рассмотрим следующие определения.

**sealed** **trait** **Resource** {

**def** goldEquiv: Int

}

**class** **Gold** **extends** **Resource** {

**def** goldEquiv = 1

}

**trait** **MaterialResource** **extends** **Resource** {

**def** goldEquiv = 50

}

**class** **Ore** **extends** **MaterialResource**

**class** **Wood** **extends** **MaterialResource**

**trait** **PreciousResource** **extends** **Resource** {

**def** goldEquiv = 200

}

**class** **Gem** **extends** **PreciousResource**

**class** **Crystal** **extends** **PreciousResource**

**class** **Sulfur** **extends** **PreciousResource**

**class** **Mercury** **extends** **PreciousResource**

Отметьте верные утверждения.

Перечисленные классы и интерфейсы не могут быть расширены в других файлах.

Интерфейс Resource может иметь более 7 реализаций (конкретных классов).

Ближайший общий надтип классов Crystal и Ore —Resource.

Ближайший общий надтип классов Wood и Mercury —PreciousResource.

Ближайший общий надтип классов Gold, Wood и Gem —Any.

Ближайший общий надтип классов Gem и Sulfur —PreciousResource.

Интерфейс Resource имеет ровно 7 реализаций (конкретных классов).

Ближайший общий надтип классов Gold, Wood и Gem —Resource.

## **Задание 6 00:03:33**

Рассмотрим следующие Scala-файлы.

Файл blog.scala:

**package** com.myblog

**package** model

**class** **Blog**(var title: String) {

**protected** **val** \_posts: ListBuffer[Post]

**def** feed = \_posts.toSeq.sortBy(\_.date).reverse

**def** publish(post: Post) {

\_posts += post

post.date = **new** Date

}

}

Файл post.scala:

**package** com.myblog

**package** model

**class** **Post**(var text: String,

var date: Date)

Отметьте верные утверждения.

Класс Post неизменяемый.

Классы пакета com.myblog видны в обоих файлах.

Поля класса Blog неизменяемые.

Класс Blog расширяет класс Post.

Класс Post расширяет класс Blog.

Классы пакета com.myblog.model видны в обоих файлах.

Поля класса Post неизменяемые.

Класс Blog неизменяемый.

## **Задание 7 00:04:37**

Рассмотрим следующие интерфейсы.

*// Многоугольник*

**trait** **Polygon** {

*// Количество сторон*

**def** sides: Int

*// Периметр*

**def** perimeter: Double

*// Площадь*

**def** area: Double

}

*// Правильный выпуклый многоугольник*

**trait** **RegularConvexNgon**

**extends** Polygon {

*// Длина стороны*

**def** side: Double

*// Радиус описанной окружности*

**def** circumradius = side / (2 \* math.sin(math.Pi / sides))

*// Радиус вписанной окружности*

**def** apothem = circumradius \* math.cos(math.Pi / sides)

*// Периметр*

**def** perimeter = sides \* side

*// Площадь*

**def** area = sides \* side \* apothem / 2

}

Какие методы необходимо реализовать конкретному классу, расширяющему интерфейс RegularConvexNgon?

side

angle

circumradius

sides

area

perimeter

apothem

Какие методы необходимо реализовать конкретному классу, расширяющему интерфейс Polygon?

perimeter

sides

circumradius

area

side

apothem

angle

Отметьте корректные варианты реализации класса EquilateralTriangle, моделирующего равносторонний треугольник.

Вариант 1:

**class** **EquilateralTriangle**(val side: Double)

**extends** RegularConvexNgon {

**def** sides = 3

}

Вариант 2:

**class** **EquilateralTriangle**(val side: Double,

val sides: Int = 3)

**extends** RegularConvexNgon

Вариант 3:

**class** **EquilateralTriangle**(var side: Double)

**extends** RegularConvexNgon {

**val** sides: Int = 3

}

## **Задание 8 00:02:23**

Рассмотрим следующие Scala-файлы.

Файл customer.scala:

**package** com.myshop

**package** model

**import** collection.mutable.ListBuffer

**class** **Customer**(val name: String,

val address: String) {

**protected** **val** \_orders = **new** ListBuffer[Order]

**def** orders = \_orders.toSeq

**def** add(order: Order) {

\_orders += order

}

}

Файл order.scala:

**package** com.myshop.model

**class** **Order**(val customer: Customer,

val status: String,

val date: Date)

Отметьте верные утверждения.

Классы пакета com.myshop видны в обоих файлах.

Класс Order расширяет класс Customer.

Поля класса Order неизменяемые.

Класс Customer расширяет класс Order.

Классы пакета com.myshop.model видны в обоих файлах.

Поля класса Customer неизменяемые.

Класс Order неизменяемый.

Класс Customer неизменяемый.

## **Задание 9 00:02:36**

Рассмотрим следующие определения.

*// Коробка*

**trait** **Box** {

**def** width: Double

**def** height: Double

**def** depth: Double

**final** **def** volume = width \* height \* depth

}

*// Спичка*

**class** **Match**

*// Спичечный коробок*

**class** **MatchBox** **extends** **Box** {

**def** width = 0.035

**def** height = 0.012

**def** depth = 0.05

**protected** **var** \_count = 50

**def** getMatch(): Option[Match] =

**if** (\_count > 0) {

\_count -= 1

Some(**new** Match)

} **else** None

}

Отметьте верные утверждения.

Метод volume не может быть переопределен в подтипахBox.

Метод volume можно переопределить в MatchBox.

Класс MatchBox неизменяемый.

MatchBox расширяет Box.

Все методы интерфейса Box абстрактны.

Методы класса MatchBox не имеют сторонних эффектов.

Интерфейс Box является параметризованным.

Box расширяет MatchBox.