This Scala notebook uses *BeakerX*, a Two Sigma Open Source project that enhances Jupyter.

http://beakerx.com/ (http://beakerx.com/)

```
In [1]:  scala.util.Properties.versionMsg

Out[1]:  Scala library version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
```

As an example of using Scala traits for mix-in functionality, we'll implement some simple classes representing Magic the Gathering creatures with mix-in abilities.

Cards and other images are linked from https://magic.wizards.com/ (https://magic.wizards.com/), © Wizards of the Coast LLC.

Mana

```
In [2]: object Color {
            case object White extends Color
            case object Blue extends Color
            case object Black extends Color
            case object Red extends Color
            case object Green extends Color
            case object Generic extends Color
            case object Colorless extends Color
        }

        sealed abstract class Color
```

```
Out[2]: defined object Color
        defined class Color
```

Card States

```
In [3]: object State {
            case object Library extends State
            case object Hand extends State
            case object Tapped extends State
            case object Untapped extends State
            case object Graveyard extends State
            case object Exile extends State
        }

        sealed abstract class State
```

```
Out[3]: defined object State
        defined class State
```

Cards

Aether Meltdown
Attune with
Die Young
Harnessed Ligh
Thriving Turtle
Creature — Turtle
When Thriving Turtle enters the battlefield, you get (two energy counters).
Whenever Thriving Turtle attacks, you may pay. If you do, put a +1/+1 counter on it.
Aether shapes everything on Kaladesh, from the sky to the sea.
0/3
Instant
Sorcery
Enchantment

```
In [4]: abstract class Card {
            /** The mana cost to play this card */
            def cost: Map[Color, Int]

            /** State card enters after it is played */
            def postPlayState: State
        }
Out[4]: defined class Card
```

Sorceries

```
In [5]: abstract class Sorcery extends Card {
            /** Sorceries are discarded after being played */
            def postPlayState = State.Graveyard
        }
Out[5]: defined class Sorcery
```

Creatures

In [6]:
```
abstract class Creature(
    /** Creature's summoning cost */
    val cost: Map[Color, Int],
    /** Creature's attack power */
    val power: Int,
    /** Creature's defensive toughness */
    val toughness: Int) extends Card {

    /**
     * Check if other creature can block this creature's attacks.
     * By default, any creature can block any other creature.
     */
    def isBlockableBy(other: Creature): Boolean = true

    /**
     * Check if this creature can attack in the given state.
     * By default, creatures can attack iff untapped.
     */
    def canAttack(currentState: State): Boolean =
        currentState == State.Untapped

    /**
     * Early (first-strike) combat damage.
     * By default, creatures don't deal early damage.
     */
    def earlyCombatDamange: Option[Int] = None

    /**
     * Normal (non-first-strike) combat damage.
     * By default, this is equal to the creature's power.
     */
    def normalCombatDamange: Option[Int] = Some(power)

    /** Creatures enter the battlefield with summoning sickness */
    def postPlayState = State.Tapped
}
```

Out[6]:
```
defined class Creature
```

Abilities

Reach & Flying

In [7]:
```
trait Reach { this: Creature => }
```

Out[7]:
```
defined trait Reach
```

In [8]:
```
trait Flying { this: Creature =>
    /**
     * Flying creatures can only be blocked by
     * other Flying creatures or by creatures with Reach.
     */
    override def isBlockableBy(other: Creature): Boolean = other match {
        case (_: Flying) | (_: Reach) => true
        case _ => false
    }
}
```
Out[8]:
```
defined trait Flying
```

Defender

In [9]:
```
trait Defender { this: Creature =>
    /** Defenders can never attack */
    override def canAttack(currentState: State) = false
}
```
Out[9]:
```
defined trait Defender
```

First-strike

In [10]:
```
trait FirstStrike { this: Creature =>
    /** Creatures with first-strike deal damage early. */
    override def earlyCombatDamange: Option[Int] = Some(power)

    /** Creatures with first-strike deal no normal damage. */
    override def normalCombatDamange: Option[Int] = None
}
```
Out[10]:
```
defined trait FirstStrike
```

Double-strike

In [11]:
```
trait DoubleStrike { this: Creature =>
    /** Creatures with double-strike deal early AND normal damage. */
    override def earlyCombatDamange: Option[Int] = Some(power)
}
```
Out[11]:
```
defined trait DoubleStrike
```

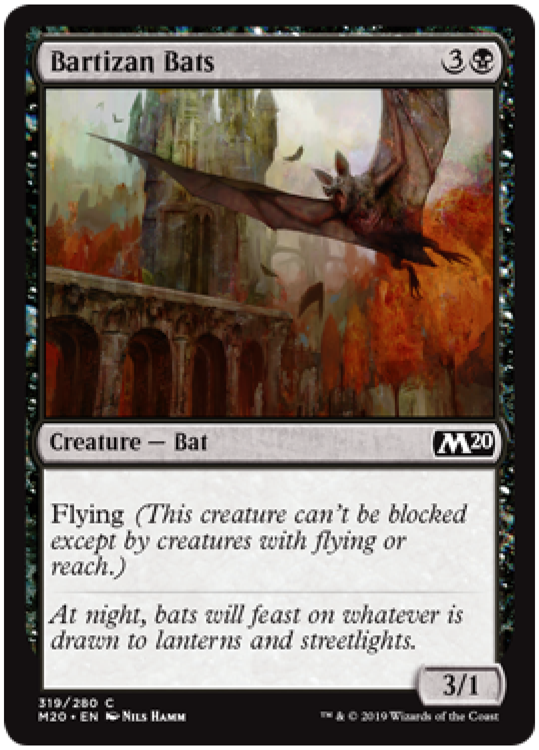
Haste

```
In [12]: trait Haste { this: Creature =>
             /** Creatures with haste enter battle untapped. */
             override def postPlayState = State.Untapped
         }

Out[12]: defined trait Haste
```

Implementing Cards with Abilities

Bartizan Bats

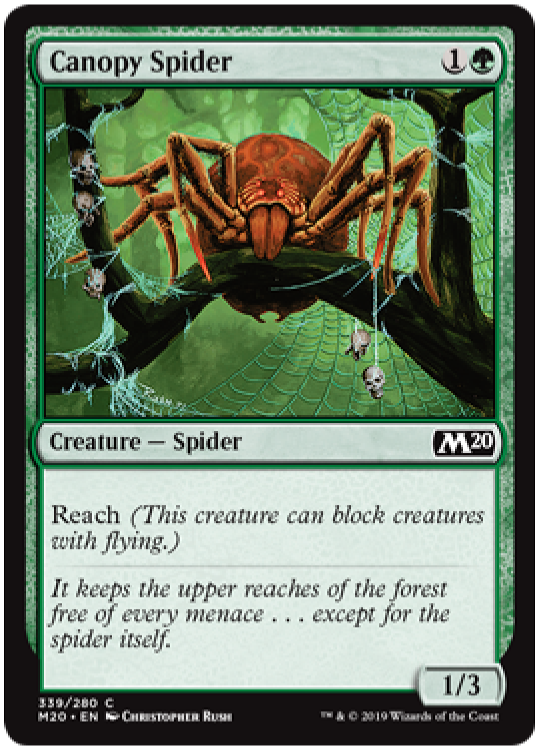

Bartizan Bats
3
Creature — Bat
M20
Flying (This creature can't be blocked except by creatures with flying or reach.)
At night, bats will feast on whatever is drawn to lanterns and streetlights.
3/1
319/280 C
M20 • EN Nils Hamm
™ & © 2019 Wizards of the Coast

(https://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=469872)

```
In [13]: case object BartizanBats extends Creature(Map(Color.Generic->3, Color.Black->1
         ), 3, 1) with Flying

Out[13]: BartizanBats
```

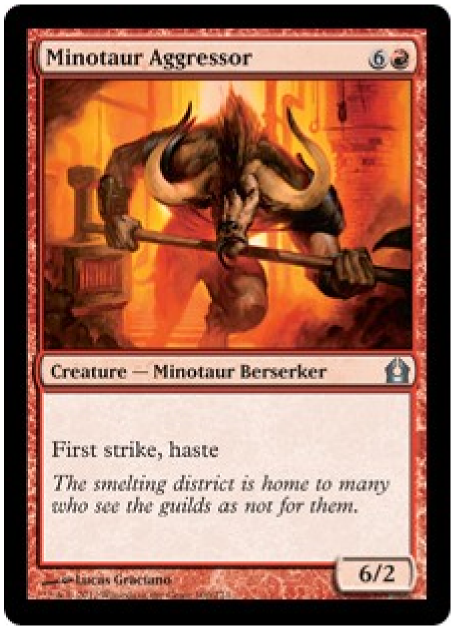
Canopy Spider


Canopy Spider
1
Creature — Spider
M20
Reach (This creature can block creatures with flying.)
It keeps the upper reaches of the forest free of every menace . . . except for the spider itself.
1/3
339/280 C
M20 • EN Christopher Rush
™ & © 2019 Wizards of the Coast
(https://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=469892)

In [14]:
```
case object CanopySpider extends Creature(Map(Color.Generic->1, Color.Green->1
), 1, 3) with Reach
```

Out[14]:
```
CanopySpider
```

Minotaur Aggressor

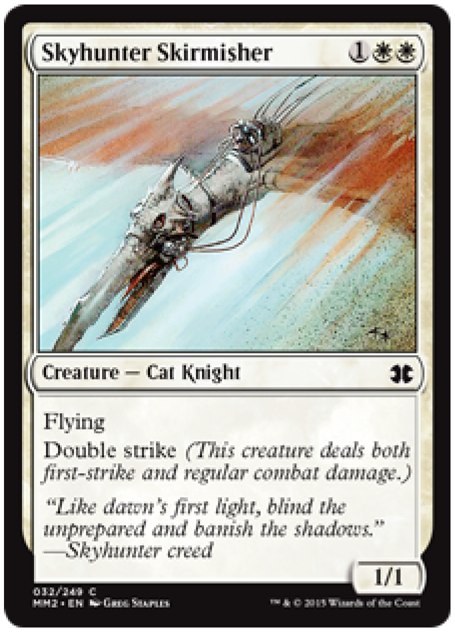

Minotaur Aggressor
6
Creature — Minotaur Berserker
First strike, haste
The smelting district is home to many who see the guilds as not for them.
6/2
Lucas Graciano

(https://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=270800)

```
In [15]: case object MinotaurAggressor extends Creature(Map(Color.Generic->6, Color.Red
         ->1), 6, 2) with FirstStrike with Haste

Out[15]: MinotaurAggressor
```

Skyhunter Skirmisher

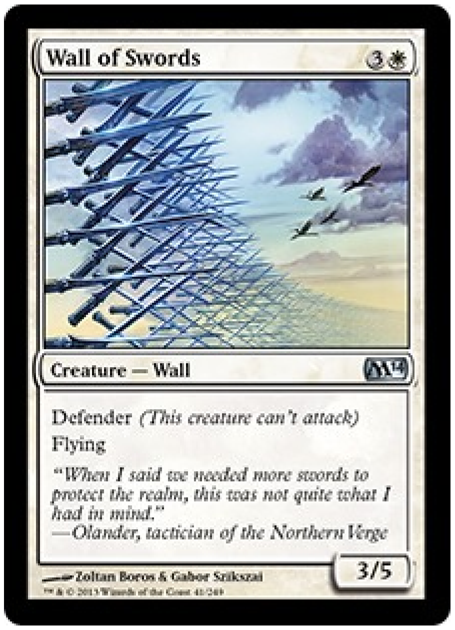

Skyhunter Skirmisher
1
Creature — Cat Knight
Flying
Double strike (This creature deals both first-strike and regular combat damage.)
"Like dawn's first light, blind the unprepared and banish the shadows."
—Skyhunter creed
1/1
032/249 C
MM2 • EN Greg Staples
™ & © 2015 Wizards of the Coast

(https://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=397835)

In [16]:
```
case object SkyhunterSkirmisher extends Creature(Map(Color.Generic->1, Color.White->2), 1, 1) with Flying with DoubleStrike
```

Out[16]:
```
SkyhunterSkirmisher
```

Wall of Swords


Wall of Swords
Creature — Wall
Defender (This creature can't attack)
Flying
"When I said we needed more swords to protect the realm, this was not quite what I had in mind."
—Olander, tactician of the Northern Verge
3/5
Zoltan Boros & Gabor Szikszai

(https://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=29772)

```
In [17]: case object WallOfSwords extends Creature(Map(Color.Generic->3, Color.White->1
         ), 3, 5) with Defender with Flying

Out[17]: WallOfSwords
```

Woodland Druid

Woodland Druid
Creature — Druid
What is a tree?
Slow hard thought.
What is a flower?
Swift soft flesh.
What is a druid?
Both.
Illus. Rick Farrell
1/2

(https://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=370697)

```
In [18]:  case object WoodlandDruid extends Creature(Map(Color.Green->1), 1, 2)

Out[18]:  WoodlandDruid
```

Allowed to Block?

```
In [19]:  SkyhunterSkirmisher isBlockableBy MinotaurAggressor  // Flying vs Normal

Out[19]:  false

In [20]:  SkyhunterSkirmisher isBlockableBy CanopySpider  // Flying vs Reach

Out[20]:  true

In [21]:  WoodlandDruid isBlockableBy WallOfSwords  // Flying vs Flying

Out[21]:  true
```

```
In [22]: WoodlandDruid isBlockableBy BartizanBats  // Normal vs Flying

Out[22]: true

In [23]: CanopySpider isBlockableBy WoodlandDruid  // Reach vs Normal

Out[23]: true

In [24]: MinotaurAggressor isBlockableBy CanopySpider  // Normal vs Reach

Out[24]: true
```