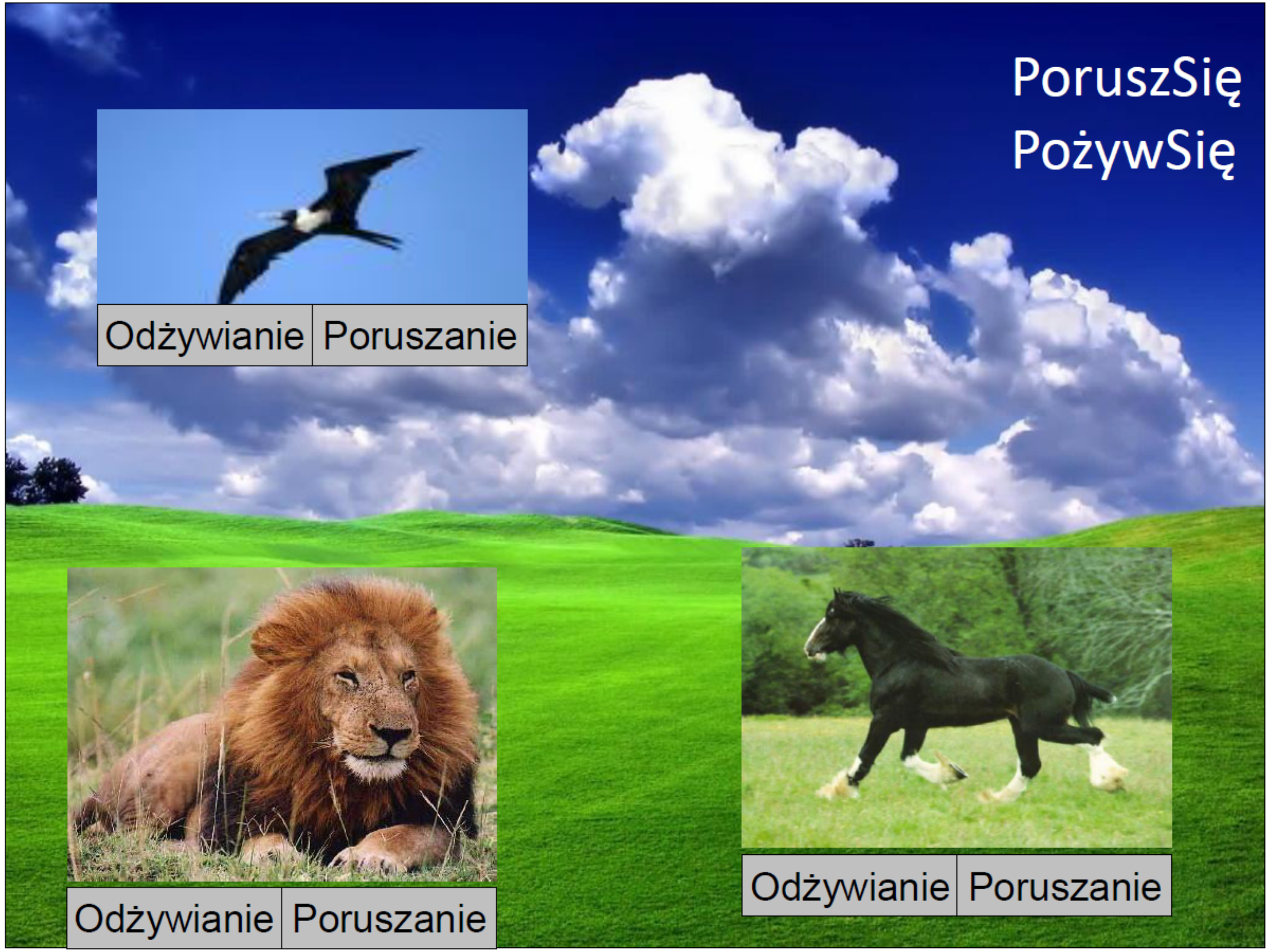
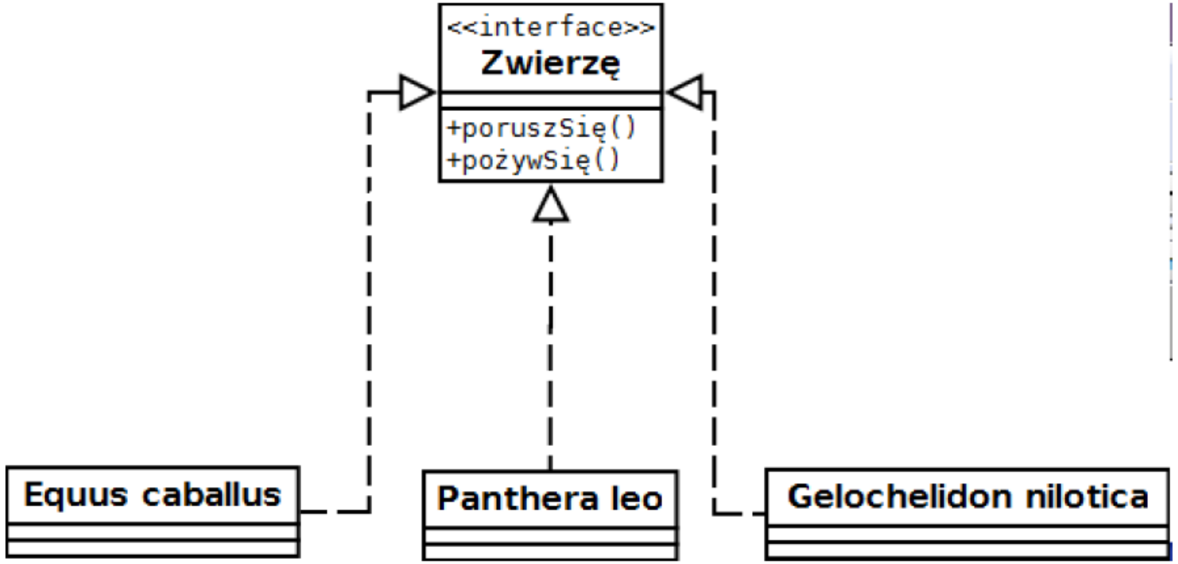
Strategy

Discovering design patterns



```
public class ZwierzeUtils
{
   public static void ZalatwSprawe(...)
   {
      if...
      if...
      ...
   }
}
```



<<interface>>

Zwierzę

+poruszSię()

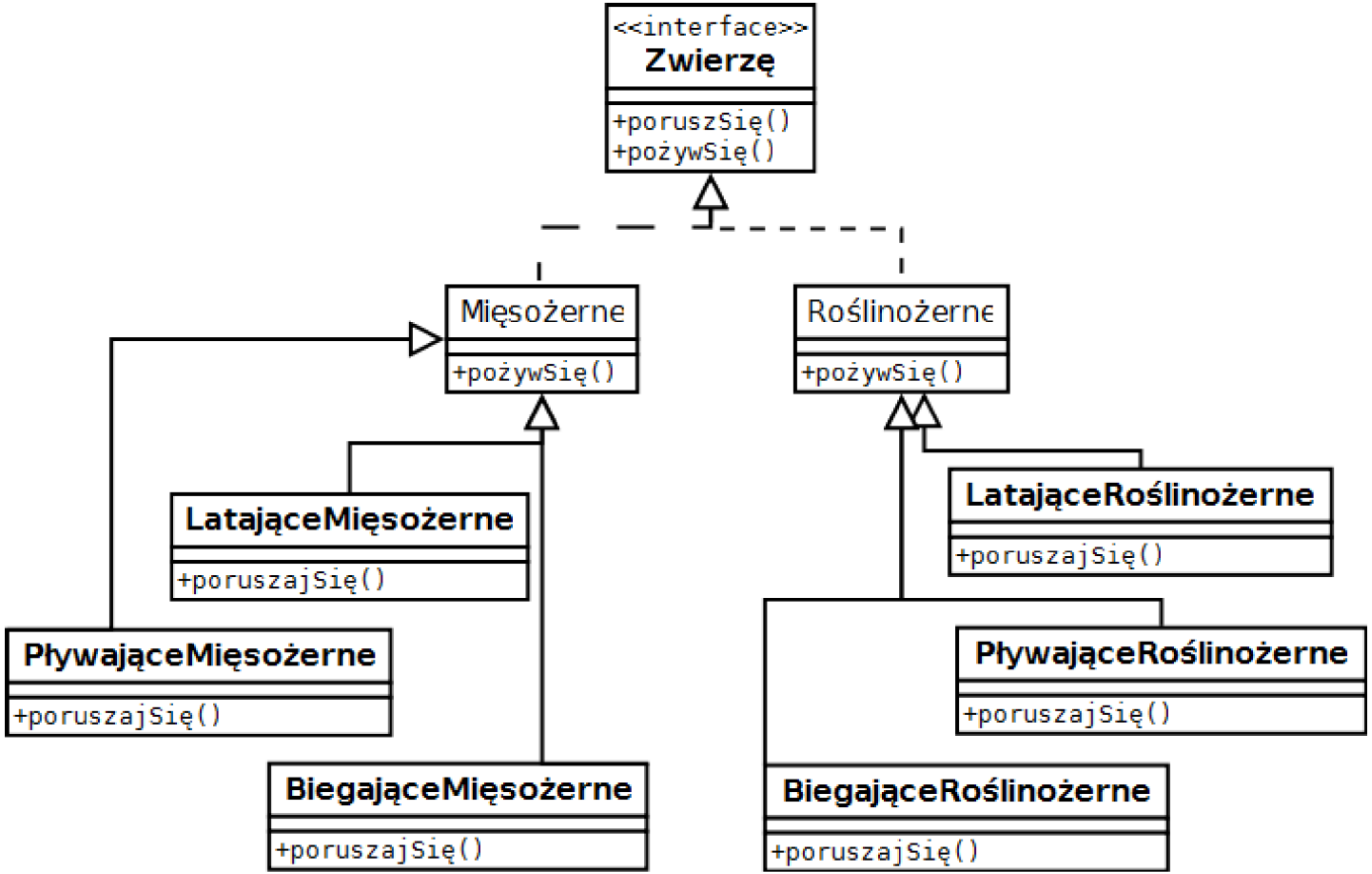
+pożywSię()

Equus caballus

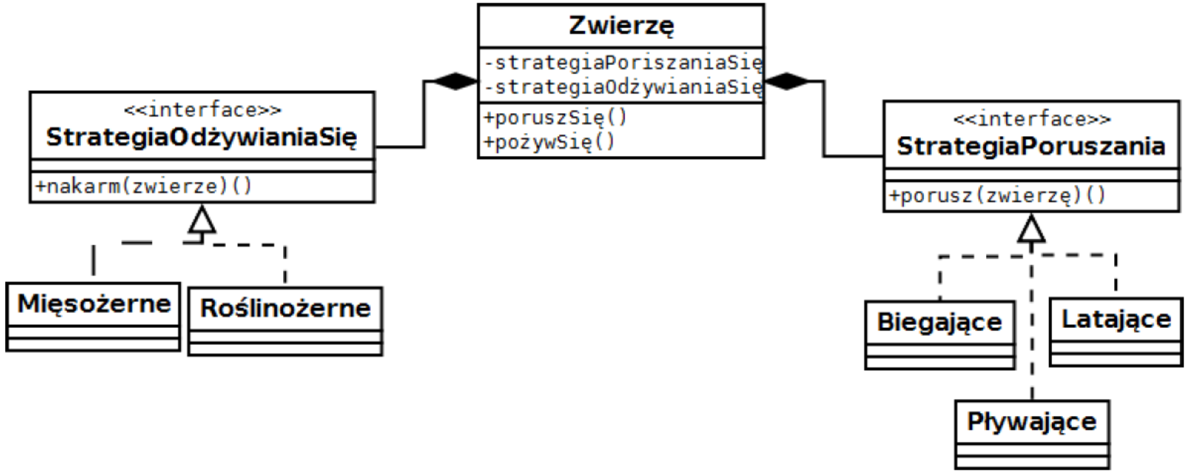
Panthera leo

Gelochelidon nilotica

Combinatorial explosion



Eureka!



```
public void PoruszSie()
{
   this.StrategiaPoruszaniaSir.Porusz(this);
}
public void PozywSie()
{
   this.StrategiaOdzywianiaSie.Nakarm(this);
}
```

Strategy

Definition: "Define a family of algorithms, encapsulate each one, and make them **interchangeable**"

Strategy

```
public class Order
{
   private ITaxPolicy _taxPolicy;
   private IRebatePolicy _rebatePolicy;
   public void Submit()
   {
      ...
      _rebatePolicy.CalculateRebate(this);
      ...
      _taxPolicy.CalculateTax(this);
      ...
   }
}
```

Strategy - summary

- Context:
  - There are variations in behavior.
  - Have a common contract
- Implementation:
  - Aggregation instead of inheritance.
    - Inheritance can be useful in the hierarchy of the strategy itself.
  - Hermetization of variation beyond a stable interface.
  - In C# can be relegated to delegate/lambda as argument.
- Strengths:
  - Consistent responsibilities.
  - Open to extension without modification.
  - Applicability only to a stable "API".
- Consequences:
  - Strategies no longer have access to context fields.
  - Convenient testing of individual strategies.
  - Convenient testing of context (strategy mocking).
  - Adding/removing strategies without modifying the context.