**MAIN**

```
{
   EntityBirdAir airbird = new EntityBirdAir("AirBird");

   EntityDog dog = new EntityDog("Dog");

   dog.acceptAttackBy(airbird);

   dog.refreshAttackBy(airbird);

   dog.releaseAttackBy(airbird);
}
```

**public abstract class A_Entity<_ATTACKER_ extends I_CanAttack> implements I_Identifiable**

```
{
   private final String _id;

   private final String _defenseMethod;

   public A_Entity(final String id, final String defenseMethod)
   {
      if ((id == null) || (defenseMethod == null))
      {
         throw new NullPointerException();
      }

      if (id.isEmpty())
      {
         throw new RuntimeException("missing id");
      }

      if (defenseMethod.isEmpty())
      {
         throw new RuntimeException("missing defense method");
      }

      _id = id;
      _defenseMethod = defenseMethod;
   }

   public void acceptAttackBy(final _ATTACKER_ attacker)
   {
      if (attacker == null)
      {
         throw new NullPointerException();
      }

      attacker.beginAttackOn_(this);

      System.out.println(getID_() + " is being attacked by " + attacker.getID_() + " by " +
                         attacker.getAttackMethod_());
   }

   public String getDefenseMethod()
   {
      return _defenseMethod;
   }
   @Override
   public String getID_()
   {
      return _id;
   }

   public void refreshAttackBy(final _ATTACKER_ attacker)
   {
      if (attacker == null)
      {
         throw new RuntimeException();
      }

      attacker.updateAttackOn_(this);

      System.out.println(getID_() + " is defending itself against " + attacker.getID_() + " by " +
                         getDefenseMethod());
   }

   public void releaseAttackBy(final _ATTACKER_ attacker)
   {
      if (attacker == null)
      {
         throw new NullPointerException();
      }

      attacker.endAttackOn_(this);

      System.out.println(getID_() + " is no longer being attacked by " + attacker.getID_() + " by "
                         + attacker.getAttackMethod_());
   }
}

public abstract class A_EntityAttacking<_ATTACKER_ extends I_CanAttack>
   extends A_Entity<_ATTACKER_> implements I_CanAttack
{
   private final String _attackMethod;

   public A_EntityAttacking(final String id, final String attackMethod,
                            final String defenseMethod)
   {
      super(id, defenseMethod);

      if (attackMethod == null)
      {
         throw new NullPointerException();
      }

      _attackMethod = attackMethod;
   }

   @Override
   public void beginAttackOn_(final A_Entity<?> attackee)
   {
      assert (attackee != null);

      System.out.println(getID_() + " has begun attacking " + attackee.getID_() + " by " +
                         getAttackMethod_());
   }
   @Override
   public void endAttackOn_(final A_Entity<?> attackee)
   {
      assert (attackee != null);

      System.out.println(getID_() + " has finished attacking " + attackee.getID_() + " by " +
                         getAttackMethod_());
   }

   @Override
   public String getAttackMethod_()
   {
      return _attackMethod;
   }

   @Override
   public void updateAttackOn_(final A_Entity<?> attackee)
   {
      assert (attackee != null);

      System.out.println(getID_() + " is again attacking " + attackee.getID_() + " by " +
                         getAttackMethod_());
   }
}

public abstract class A_EntityBird<_ATTACKER_ extends I_CanAttackBird>
   extends A_EntityAttacking<_ATTACKER_>
{
   public A_EntityBird(final String id)
   {
      super(id, "chirping really annoyingly", "singing");
   }
}

public abstract class A_EntityTree<_ATTACKER_ extends I_CanAttackTree>
   extends A_Entity<_ATTACKER_>
{
   public A_EntityTree(final String id, final String defenseMethod)
   {
      super(id, defenseMethod);
   }
}

public class EntityBirdAir extends A_EntityBird<I_CanAttackBirdAir>
   implements I_CanAttackBirdAir, I_CanAttackBirdGround, I_CanAttackBirdWater,
      I_CanAttackDog, I_CanAttackTree
{
   public EntityBirdAir(final String id)
   {
      super(id);
   }
   @Override
   public void acceptAttackBy(final I_CanAttackBirdAir attacker)
   {
      super.acceptAttackBy(attacker);

      System.out.println(getID_() + " launched a bird dropping at " + attacker.getID_());
   }

   @Override
   public void releaseAttackBy(final I_CanAttackBirdAir attacker)
   {
      super.releaseAttackBy(attacker);

      System.out.println(getID_() + " flipped " + attacker.getID_() + " the bird and flew away");
   }
}

public class EntityBirdGround extends A_EntityBird<I_CanAttackBirdGround>
   implements I_CanAttackBirdGround, I_CanAttackDog, I_CanAttackTree
{
   public EntityBirdGround(final String id)
   {
      super(id);
   }
}

public class EntityBirdWater extends A_EntityBird<I_CanAttackBirdWater>
   implements I_CanAttackBirdWater, I_CanAttackFish
{
   public EntityBirdWater(final String id)
   {
      super(id);
   }
}

public class EntityClam extends A_EntityAttacking<I_CanAttackClam> implements I_CanAttackFish
{
   public EntityClam(final String id)
   {
      super(id, "gomping", "making pearls");
   }
}

public class EntityDog extends A_EntityAttacking<I_CanAttackDog>
   implements I_CanAttackBirdGround, I_CanAttackClam, I_CanAttackDog, I_CanAttackTree
{
   public EntityDog(final String id)
   {
      super(id, "biting", "barking");
   }
   @Override
   public void beginAttackOn_(final A_Entity<?> attackee)
   {
      System.out.println(attackee.getID_() + " bared its teeth, raised its tail, and started
                         growling at " + getID_());

      super.beginAttackOn_(attackee);
   }

   @Override
   public void endAttackOn_(final A_Entity<?> attackee)
   {
      super.endAttackOn_(attackee);

      System.out.println(attackee.getID_() + " closed its mouth and walked away looking back at " +
                         getID_());
   }

   @Override
   public void updateAttackOn_(final A_Entity<?> attackee)
   {
      super.endAttackOn_(attackee);

      System.out.println(attackee.getID_() + " made nasty noises at " + getID_());
   }
}

public class EntityFish extends A_EntityAttacking<I_CanAttackFish>
   implements I_CanAttackBirdWater, I_CanAttackClam, I_CanAttackFish
{
   public EntityFish(final String id)
   {
      super(id, "nibbling", "blowing bubbles");
   }
}

public class EntityGhost extends A_EntityAttacking<I_CanAttackGhost>
   implements I_CanAttackBirdAir, I_CanAttackBirdGround, I_CanAttackDog, I_CanAttackTree
{
   public EntityGhost(final String id)
   {
      super(id, "making spooking noises", "floating around");
   }
}

public class EntityTreeOak extends A_EntityTree<I_CanAttackTreeOak>
{
   public EntityTreeOak(final String id)
   {
      super(id, "dropping acorns");
   }
}
```

```
public class EntityTreePine extends A_EntityTree<I_CanAttackTreePine>
{
   public EntityTreePine(final String id)
   {
      super(id, "dropping pine needles");
   }
}
```