```
abstract class Critter{
  abstract public void makeNoise();
}
class Mouse extends Critter{
  public void makeNoise(){squeek();}
  public void squeek(){S.O.P("SQUEEK!");}
}
class Cat extends Critter{
  public void makeNoise(){meow();}
  public void meow(){S.O.P("MEOW!");}
}
class driver{
  public static void main(String [] args){
    Critter bugger = new Cat();  //normal polymorphism
    Critter spots = new Mouse();  //normal polymorphism
    Critter meaty = new Critter();  //cannot instantiate abstract class
    bugger.meow();   //no method meow found in class Critter
    bugger.squeek();  //no method squeek found in class Critter
    bugger.makeNoise();  //normal call OK
    Cat myCat = (Cat)bugger;  //explicit downcasting
    myCat.meow();  //normal call OK
    myCat.makeNoise();  //normal call OK
  }
}
```

```
class Box{
  private boolean open=false;
  public static boxes=0;  //bad practice public instance!
  public Box(){boxes++;}
  public string examineContents(){…}
  public static String describeBox(){
    return “Boxes are 6 sided cardboard things.”;
  }
  public String toString(){
    return “This box is ” + (open?”open”:”closed”);  //Ternary operator
  }
}
class Driver{
  public static void main(String [] args){
    Box b = new Box();
    S.O.P(b.boxes);  //instances can see static variables
    S.O.P(Box.boxes);  //static references can see static variables
    S.O.P(b);   //instances can get at instance methods
    S.O.P(Box);  //static references cannot get at instance methods!
  }
}
```