Data Clumps

Data clumps are multiple parameters that appear repeatedly in several method signatures.

Data clumps are a problem because a change to one clump necessitates the same change in several places.

To correct this problem, the duplicated parameters should be encapsulated into objects.

For instance, if the integers x, y, and z appear as parameters to several methods, they could be replaced by a new 3DPoint class.

Feature Envy

Feature envy occurs when code relies heavily on data and methods from other classes, more that those from its own class.

Feature envy is a problem because the responsibilities of a class should be contained in the class itself.

To correct this problem, code that exhibits feature envy should be moved into the appropriate class.

Message Chain

A message chain is a long series of method calls on the same object.

Message chains are a problem because the client code becomes tightly coupled with the object's structure.

To correct this problem, parts of the message chains can be hidden in various objects, or methods can be moved closer to clients.

For instance, this.getFigure().getPolygon(0).getPoint(0).x() might be replaced by this.farthestLeft().

Switch Statement

A switch statement is a language feature.

Switch statements are a problem because, often, they are duplicated many places in code.

To correct this problem, switch statements may be replaced with polymorphism.

Feature Envy: Example

```
class Figure{
void add(Point p1, Point p2){
   return new Point(p1.x+p2.x,p1.y+p2.y);
}
…
```

Data Clumps: Example

```
void add(int x, int y, int z){…}
void remove(int x, int y, int z){…}
void shift(int x, int y, int z){…}
void with(int x, int y, int z){…}
```

Switch Statement: Example

```
String pointString(){
 switch(getPoint().getType()){
  case TWO_DIMENSIONAL: return “2d”;
  case THREE_DIMENSIONAL: return “3d”;
  case FOUR_DIMENSIONAL: return “4d”;
  default: return “unknown”;}
}
```

Message Chain: Example

```
this.getFigure().getPolygon(0).getPoint(0).x()
```

Typecast

A typecast is where an object of one type is cast to an object of another type.

Typecasts are a problem because illegal casts can be written that will result in runtime errors.

Typecasts can sometimes be replaced with generics.

InstanceOf

An InstanceOf is a language feature.

InstanceOf is a problem because they are often duplicated many places in code.

InstanceOf checks can sometimes be replaced with polymorphism.

Large Method

A large method is a method that is long or does too many things.

Large methods are a problem because they can be difficult to understand.

Large methods can often be broken down into smaller methods.

Large Class

A large class is a class that is too long or has too many responsibilities.

Large classes are a problem because they can be difficult to understand.

Large classes can often be broken down into smaller classes.

InstanceOf: Example

```
String pointString(Point p){
 if(p instanceof 2DPoint){
   return “2d”;
 }else if(p instanceof 3DPoint){
   return “3d”;
 }else if(p instanceof 4DPoint){
   return “4d”;
 }else{
   return “unknown”;}
}
```

Typecast: Example

```
void add(Collection c){
  this.x = ((Integer)c.get(0)).intValue():
  this.y = ((Integer)c.get(1)).intValue():
  this.z = ((Integer)c.get(2)).intValue():
}
```

Large Class: Example

(some class that does not fit on this card)

Large Method: Example

```
(some method that does not fit on this card)
```