

Module 7: OO - Testing recap

CSI3105:
Software Testing

Unit? Integration? System?



Test axle unit

Test wheel unit

My software!!!

Unit? Integration? System?



My software!!!

Test wheel and axle together unit

Unit? Integration? System?



My software!!!

Test it all

Typical OO software characteristics that impact testing

- State dependent behavior
- Encapsulation
- Inheritance
- Polymorphism and dynamic binding
- Abstract and generic classes
- Exception handling

- State-dependent behaviour in object-oriented programming can significantly impact software testing
- In object-oriented programming, an object's behaviour depends not only on its internal implementation but also on its current state.
 - Variables associated with a class change when methods are called
- Different inputs or events can cause the object to transition to a different state, which may result in different behaviour

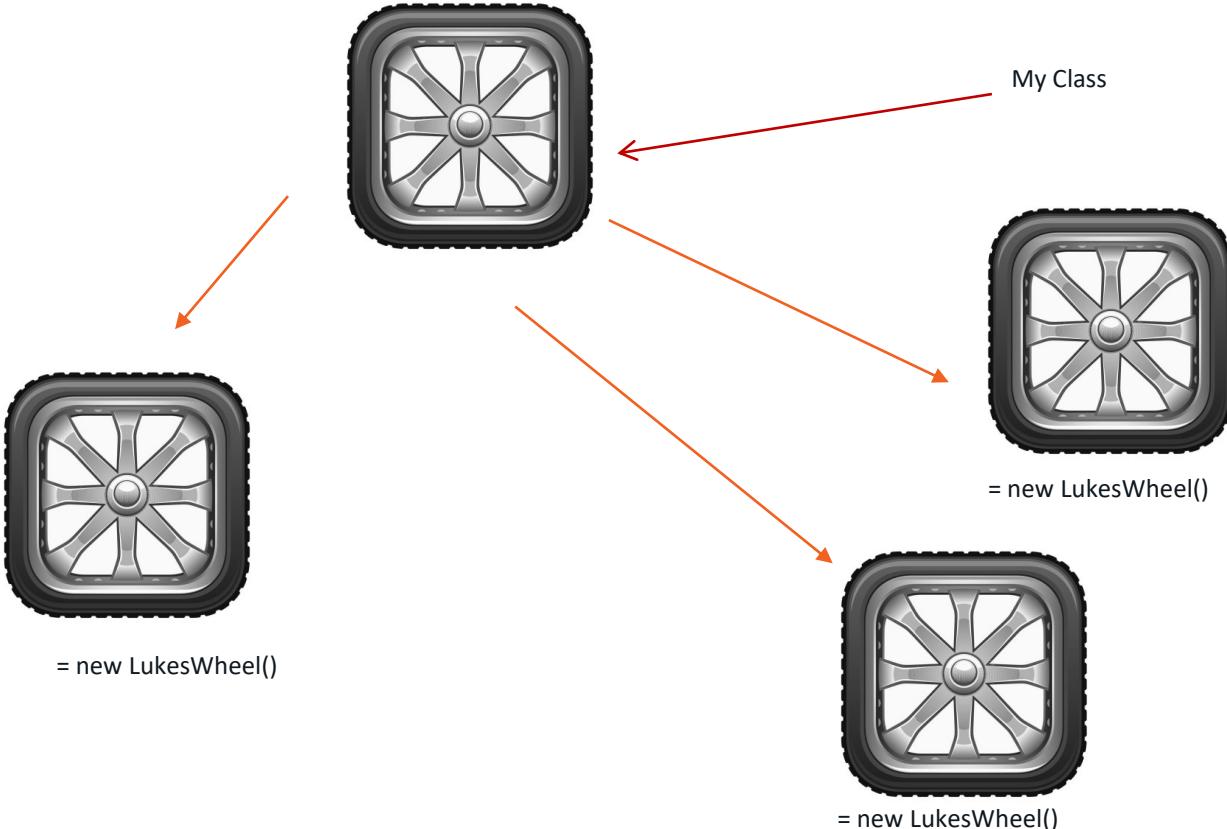


My Class

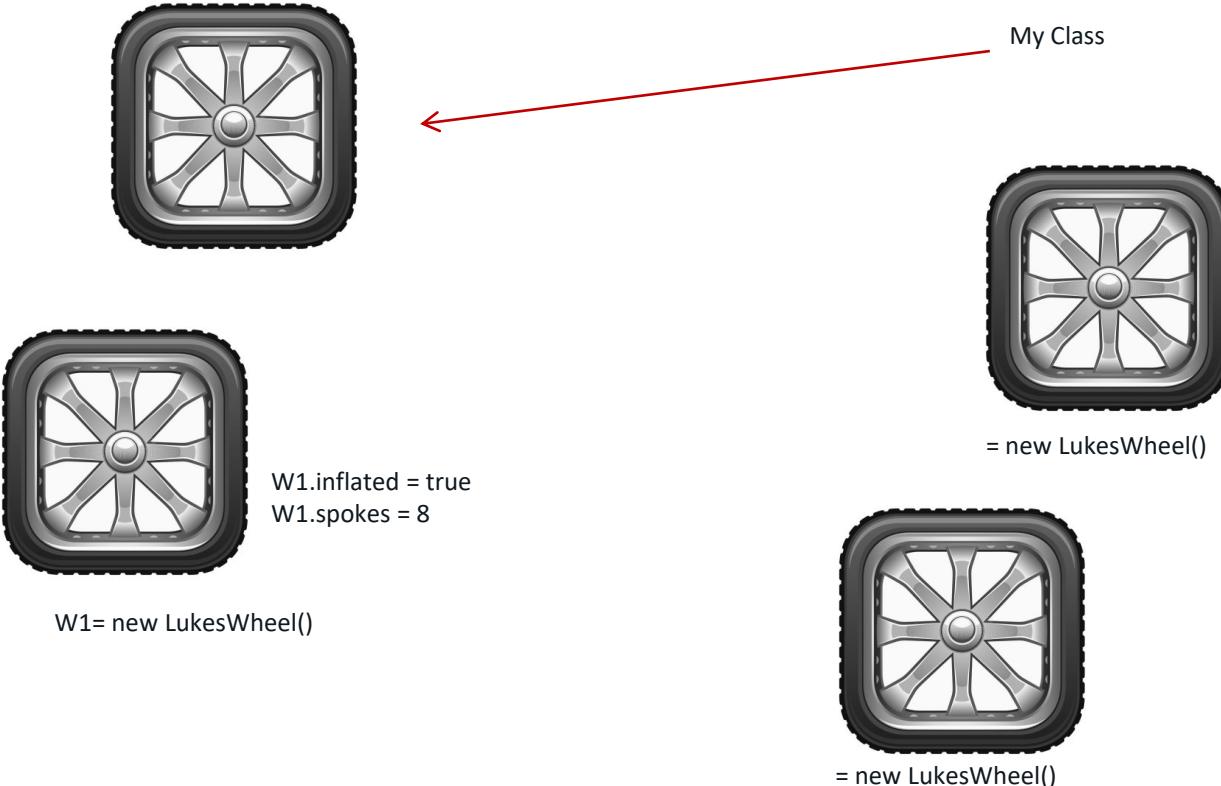


- Class... it's a thing
 - Object
 - Entity
 - Thing
- Store's data
 - Properties/ Fields
- Allows us to manipulate that data
 - Operations/ Methods

Objects as Template



Objects as Template



Objects State



My Class



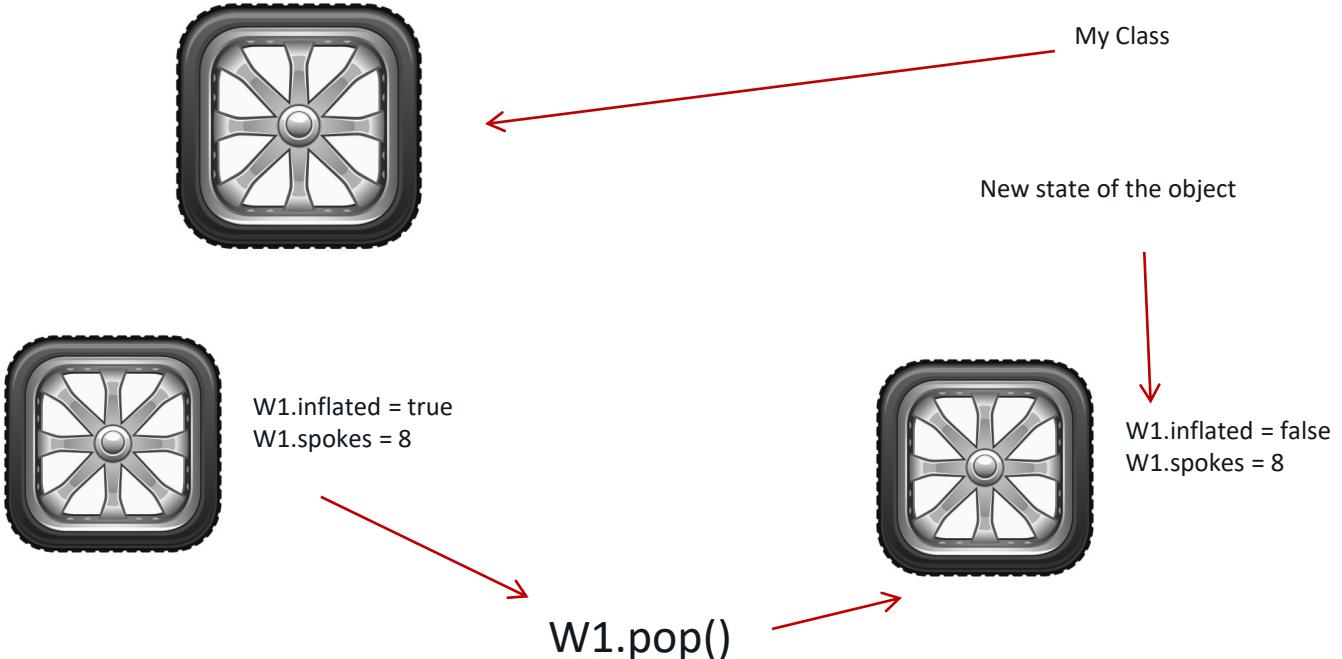
W1.inflated = true
W1.spokes = 8



State of the object

W1= new LukesWheel()

Objects State



Encapsulation

- Encapsulating or hiding variables with the private keyword, adds in some complexity
- We need to consider how to handle hidden or encapsulated fields.

Accessing the state

Intrusive approaches

- use language constructs (C++ friend classes)
- add inspector methods
- *in both cases we break encapsulation and we may produce undesired results*

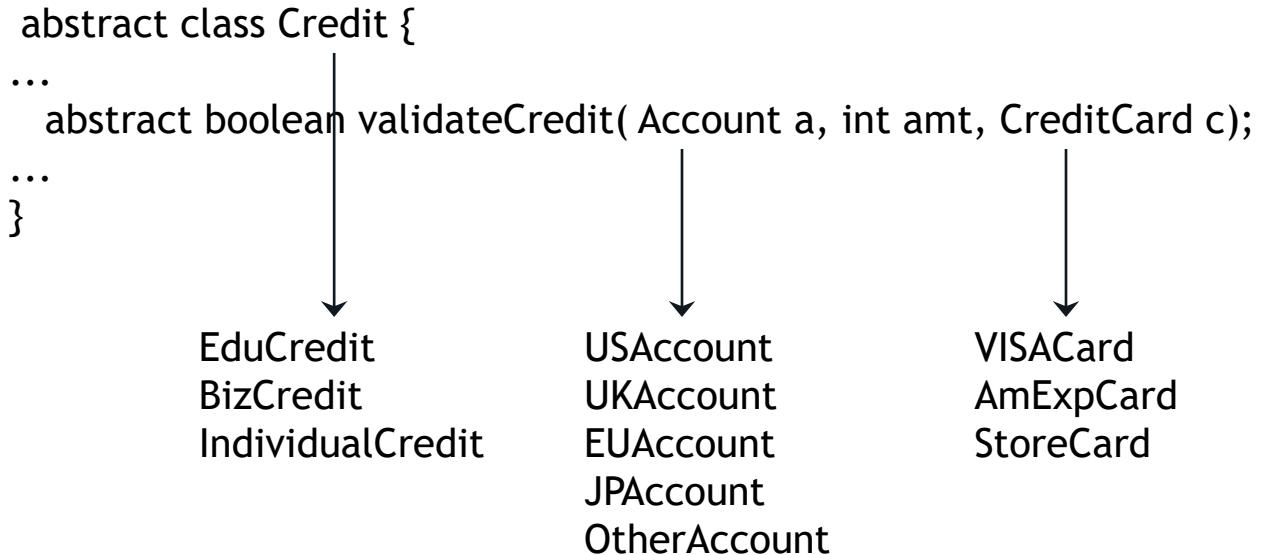
Equivalent scenarios approach:

- generate equivalent and non-equivalent sequences of method invocations
- compare the final state of the object after equivalent and non-equivalent sequences

- Inheritance
 - One class can inherit properties and behaviours from another class
 - Any issues in the base class can propagate to the derived classes.
- Testing of derived classes
 - When a class inherits from a base class, it can add or modify behaviours.
 - Test the derived classes to ensure that they behave as intended (if changed or used in a different context)
- Refactoring
 - Refactoring a base class can affect all derived classes, making it necessary to retest them all.

- Polymorphism
 - Ability of objects of different classes to be treated as if they were objects of a common superclass
 - Single method can be used to operate on objects of different classes, without needing to know the specific class of each object.
- Dynamic binding
 - A single method call may be dynamically bound to different methods depending on the state of the computation.
 - Overridden method that acts different based on the parameters it receives.
 - Tests must exercise different bindings to reveal failures that depend on a particular binding or on interactions between bindings for different calls.

Polymorphism and dynamic binding



The combinatorial problem: $3 \times 5 \times 3 = 45$ possible combinations of dynamic bindings (just for this one method!)

The combinatorial approach

Identify a set of combinations that cover all pairwise combinations of dynamic bindings

Same motivation as pairwise specification-based testing (module 8, next week)

Account	Credit	creditCard
USAccount	EduCredit	VISACard
USAccount	BizCredit	AmExpCard
USAccount	individualCredit	ChipmunkCard
UKAccount	EduCredit	AmExpCard
UKAccount	BizCredit	VISACard
UKAccount	individualCredit	ChipmunkCard
EUAccount	EduCredit	ChipmunkCard
EUAccount	BizCredit	AmExpCard
EUAccount	individualCredit	VISACard
JPAccount	EduCredit	VISACard
JPAccount	BizCredit	ChipmunkCard
JPAccount	individualCredit	AmExpCard
OtherAccount	EduCredit	ChipmunkCard
OtherAccount	BizCredit	VISACard
OtherAccount	individualCredit	AmExpCard

- Abstract classes cannot be directly instantiated
 - serve as a blueprint for other classes.
 - Testing an abstract class alone is not feasible
- Therefore, testing the concrete classes that implement the abstract class becomes important, as they provide the full implementation of the abstract class.
- Generic classes are classes/ methods that are parameterized by one or more types
 - Testing generic classes requires testing them with different types of parameters to ensure that they work correctly in all possible scenarios.

Exception handling

Exception handling

- Attempt to handle run time errors
- separate handling of error cases from the primary program logic

Try catch block etc..

- Alters the flow of code

We need to..

- Test where exceptions are thrown
- Test where exceptions are handled
- Test for unhandled exceptions

CFG - Procedural vs OOP

Differences

```

// Procedureal code example (BIG)
function calculateTotalPrice(items) {
    let total = 0;
    for (let item of items) {
        total += item.price * item.quantity;
    }
    return total;
}

```

Procedural...BIG!

```

// Object-Oriented code example (Small)
class Item {
    constructor(price, quantity) {
        this.price = price;
        this.quantity = quantity;
    }
}

class ShoppingCart {
    constructor() {
        this.items = [];
    }

    add(item) {
        this.items.push(item);
    }

    calculateTotal() {
        let total = 0;
        for (let item of this.items) {
            total += item.price * item.quantity;
        }
        return total;
    }
}

const cart = new ShoppingCart();
cart.add(new Item(100, 2));
cart.add(new Item(200, 1));
console.log(cart.calculateTotal()); // Output: 400

```

OOP...Small

Control flow graph for a class

```

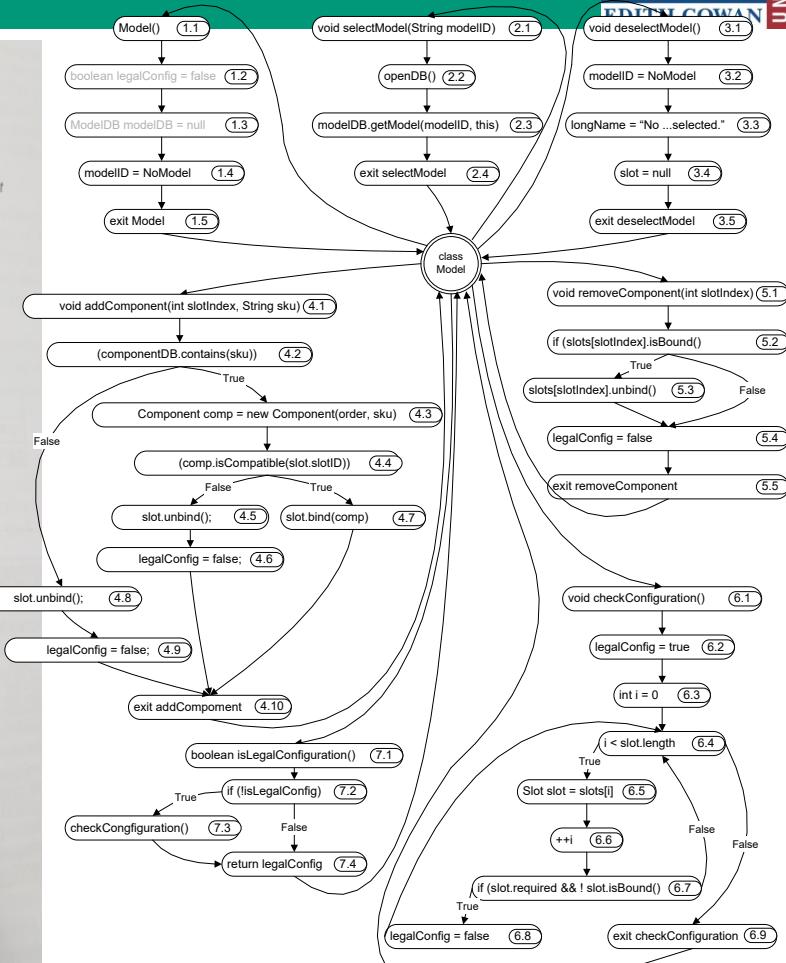
1  public class Model extends Orders.CompositeItem {
2      public String modelID; // Database key for slots
3      private int baseWeight; // Weight excluding optional components
4      private int heightCm, widthCm, depthCm; // Dimensions if boxed
5      private Slot[] slots; // Component slots
6
7      private boolean legalConfig = false; // memoized result of isLegalConfig
8      private static final String NoModel = "NO MODEL SELECTED";
9
10     ...
11
12     /** Constructor, which should be followed by selectModel */
13     public Model(Orders.Order _order) {
14         super(_order);
15         modelID = NoModel;
16     }
17
18     ...
19
20     /** Is the current binding of components to slots a legal
21      * configuration? Memo-ize the result for repeated calls */
22     public boolean isLegalConfiguration() {
23         if (!legalConfig) {
24             checkConfiguration();
25         }
26         return legalConfig;
27     }
28
29
30     /** Are all required slots filled with compatible components?
31      * It is impossible to assign an incompatible component,
32      * so just to check that every required slot is filled. */
33     private void checkConfiguration() {
34         legalConfig = true;
35         for (int i=0; i < slots.length; ++i) {
36             Slot slot = slots[i];
37             if (slot.required && ! slot.isBound()) {
38                 legalConfig = false;
39             }
40         }
41
42     }
43
44     ...
45
46     /**
47      * Bind a component to a slot.
48      * @param slotIndex Which slot (integer index)?
49      * @param sku Key to component database.
50      * Choices should be constrained by web interface, so we don't
51      * need to be graceful in handling bogus parameters.
52      */
53
54     public void addComponent(int slotIndex, String sku) {
55         Slot slot =slots[slotIndex];
56
57         if (componentDB.contains(sku)) {
58             Component comp = new Component(order, sku);
59             if (comp.isCompatible(slot.slotID)) {
60                 slot.bind(comp);
61                 // Note this cannot have made the
62                 // configuration illegal.
63             } else {
64                 slot.unbind();
65                 legalConfig = false;
66             }
67         } else {
68             slot.unbind();
69             legalConfig = false;
70         }
71
72     }
73
74     ...
75
76     /**
77      * Unbind a component from a slot.
78      */
79     public void removeComponent(int slotIndex) {
80         // assert slotIndex in 0..slots.length
81         if (slots[slotIndex].isBound()) {
82             slots[slotIndex].unbind();
83         }
84         legalConfig = false;
85
86     }
87
88     ...
89
90     /**
91      * Select a model.
92      */
93     void selectModel(String modelID) {
94         boolean legalConfig = false;
95
96         if (modelID == NoModel) {
97             exit selectModel;
98         }
99
100        ...
101
102        if (modelDB == null) {
103            exit selectModel;
104        }
105
106        if (modelID != NoModel) {
107            modelDB = modelDB.getModel(modelID, this);
108        }
109
110        if (modelDB != null) {
111            exit selectModel;
112        }
113
114        ...
115
116        if (modelDB != null) {
117            exit selectModel;
118        }
119
120    }
121
122    ...
123
124 }

```

```

public class Model extends Orders.CompositeItem {
    ...
    /* Bind a component to a slot.
     * @param slotIndex Which slot (integer index)?
     * @param sku Key to component database.
     * Choices should be constrained by web interface, so we don't
     * need to be graceful in handling bogus parameters.
     */
    public void addComponent(int slotIndex, String sku) {
        Slot slot =slots[slotIndex];
        if (componentDB.contains(sku)) {
            Component comp = new Component(order, sku);
            if (comp.isCompatible(slot.slotID)) {
                slot.bind(comp);
                // Note this cannot have made the
                // configuration illegal.
            } else {
                slot.unbind();
                legalConfig = false;
            }
        } else {
            slot.unbind();
            legalConfig = false;
        }
    }
}

```



Control flow graph for a class

```

1  public class Model extends Orders.CompositeItem {
2      public String modelID; // Database key for slots
3      private int baseWeight; // Weight excluding optional components
4      private int heightCm, widthCm, depthCm; // Dimensions if boxed
5      private Slot[] slots; // Component slots
6
7      private boolean legalConfig = false; // memoized result of isLegalConfig
8      private static final String NoModel = "NO MODEL SELECTED";
9
10     ...
11
12     /** Constructor, which should be followed by selectModel */
13     public Model(Orders.Order _order) {
14         super(_order);
15         modelID = NoModel;
16     }
17
18     ...
19
20     /** Is the current binding of components to slots a legal
21      * configuration? Memoize the result for repeated calls */
22     public boolean isLegalConfiguration() {
23         if (!legalConfig) {
24             checkConfiguration();
25         }
26         return legalConfig;
27     }
28
29     ...
30
31     /** Are all required slots filled with compatible components?
32      * It is impossible to assign an incompatible component,
33      * so just to check that every required slot is filled. */
34     private void checkConfiguration() {
35         legalConfig = true;
36         for (int i=0; i < slots.length; ++i) {
37             Slot slot = slots[i];
38             if (slot.required && !slot.isBound()) {
39                 legalConfig = false;
40             }
41         }
42     }
43
44     ...
45
46     /**
47      * Bind a component to a slot.
48      * @param slotIndex Which slot (integer index)?
49      * @param sku Key to component database.
50      * Choices should be constrained by web interface, so we don't
51      * need to be graceful in handling bogus parameters.
52      */
53
54     public void addComponent(int slotIndex, String sku) {
55         Slot slot = slots[slotIndex];
56         if (componentDB.contains(sku)) {
57             Component comp = new Component(order, sku);
58             if (comp.isCompatible(slot.slotID)) {
59                 slot.bind(comp);
60                 // Note this cannot have made the
61                 // configuration illegal.
62             } else {
63                 slot.unbind();
64                 legalConfig = false;
65             }
66         } else {
67             slot.unbind();
68             legalConfig = false;
69         }
70     }
71
72     ...
73
74     /**
75      * Unbind a component from a slot.
76      */
77     public void removeComponent(int slotIndex) {
78         // assert slotIndex in 0..slots.length
79         if (slots[slotIndex].isBound()) {
80             slots[slotIndex].unbind();
81         }
82     }
83
84     ...
85
86     /**
87      * Check if the current configuration is legal.
88      */
89     private boolean isLegalConfiguration() {
90         if (!legalConfig) {
91             return legalConfig;
92         }
93         legalConfig = false;
94     }
95
96     ...
97
98     /**
99      * Select a model for the composite item.
100     */
101    void selectModel(String modelID) {
102        if (modelID == NoModel) {
103            longName = "No ...selected."
104        }
105    }
106
107    /**
108     * Deselect the current model.
109     */
110    void deselectModel() {
111    }
112
113    /**
114     * Open the database connection.
115     */
116    void openDB() {
117    }
118
119    /**
120     * Get the model for the specified ID.
121     */
122    ModelDB getModel(modelID, this) {
123    }
124
125    ...
126
127    /**
128     * Exit the selectModel method.
129     */
130    void exit selectModel() {
131    }
132
133    /**
134     * Exit the deselectModel method.
135     */
136    void exit deselectModel() {
137    }
138
139    /**
140     * Remove a component from a slot.
141     */
142    void removeComponent(int slotIndex) {
143
144        if (slots[slotIndex].isBound()) {
145            slots[slotIndex].unbind();
146        }
147    }
148
149    /**
150     * Check the configuration.
151     */
152    void checkConfiguration() {
153
154        legalConfig = true;
155
156        int i = 0;
157
158        while (i < slots.length) {
159
160            Slot slot = slots[i];
161
162            if (slot.required && !slot.isBound()) {
163                legalConfig = false;
164            }
165
166            i++;
167
168        }
169
170    }
171
172    ...
173
174    /**
175     * Exit the addComponent method.
176     */
177    void exit addComponent() {
178    }
179
180    /**
181     * Exit the removeComponent method.
182     */
183    void exit removeComponent() {
184    }
185
186    /**
187     * Exit the checkConfiguration method.
188     */
189    void exit checkConfiguration() {
190    }
191
192    ...
193
194    /**
195     * Exit the class Model.
196     */
197    void exit Model() {
198    }
199
200    ...
201
202    /**
203     * Exit the class Model.
204     */
205    void exit class Model() {
206    }
207
208    ...
209
210    /**
211     * Exit the class Model.
212     */
213    void exit class Model() {
214    }
215
216    ...
217
218    /**
219     * Exit the class Model.
220     */
221    void exit class Model() {
222    }
223
224    ...
225
226    /**
227     * Exit the class Model.
228     */
229    void exit class Model() {
230    }
231
232    ...
233
234    /**
235     * Exit the class Model.
236     */
237    void exit class Model() {
238    }
239
240    ...
241
242    /**
243     * Exit the class Model.
244     */
245    void exit class Model() {
246    }
247
248    ...
249
250    /**
251     * Exit the class Model.
252     */
253    void exit class Model() {
254    }
255
256    ...
257
258    /**
259     * Exit the class Model.
260     */
261    void exit class Model() {
262    }
263
264    ...
265
266    /**
267     * Exit the class Model.
268     */
269    void exit class Model() {
270    }
271
272    ...
273
274    /**
275     * Exit the class Model.
276     */
277    void exit class Model() {
278    }
279
280    ...
281
282    /**
283     * Exit the class Model.
284     */
285    void exit class Model() {
286    }
287
288    ...
289
290    /**
291     * Exit the class Model.
292     */
293    void exit class Model() {
294    }
295
296    ...
297
298    /**
299     * Exit the class Model.
300     */
301    void exit class Model() {
302    }
303
304    ...
305
306    /**
307     * Exit the class Model.
308     */
309    void exit class Model() {
310    }
311
312    ...
313
314    /**
315     * Exit the class Model.
316     */
317    void exit class Model() {
318    }
319
320    ...
321
322    /**
323     * Exit the class Model.
324     */
325    void exit class Model() {
326    }
327
328    ...
329
330    /**
331     * Exit the class Model.
332     */
333    void exit class Model() {
334    }
335
336    ...
337
338    /**
339     * Exit the class Model.
340     */
341    void exit class Model() {
342    }
343
344    ...
345
346    /**
347     * Exit the class Model.
348     */
349    void exit class Model() {
350    }
351
352    ...
353
354    /**
355     * Exit the class Model.
356     */
357    void exit class Model() {
358    }
359
360    ...
361
362    /**
363     * Exit the class Model.
364     */
365    void exit class Model() {
366    }
367
368    ...
369
370    /**
371     * Exit the class Model.
372     */
373    void exit class Model() {
374    }
375
376    ...
377
378    /**
379     * Exit the class Model.
380     */
381    void exit class Model() {
382    }
383
384    ...
385
386    /**
387     * Exit the class Model.
388     */
389    void exit class Model() {
390    }
391
392    ...
393
394    /**
395     * Exit the class Model.
396     */
397    void exit class Model() {
398    }
399
399    ...
400
401    /**
402     * Exit the class Model.
403     */
404    void exit class Model() {
405    }
406
407    ...
408
409    /**
410     * Exit the class Model.
411     */
412    void exit class Model() {
413    }
414
415    ...
416
417    /**
418     * Exit the class Model.
419     */
420    void exit class Model() {
421    }
422
423    ...
424
425    /**
426     * Exit the class Model.
427     */
428    void exit class Model() {
429    }
430
431    ...
432
433    /**
434     * Exit the class Model.
435     */
436    void exit class Model() {
437    }
438
439    ...
440
440    /**
441     * Exit the class Model.
442     */
443    void exit class Model() {
444    }
445
446    ...
447
448    /**
449     * Exit the class Model.
450     */
451    void exit class Model() {
452    }
453
454    ...
455
456    /**
457     * Exit the class Model.
458     */
459    void exit class Model() {
460    }
461
462    ...
463
464    /**
465     * Exit the class Model.
466     */
467    void exit class Model() {
468    }
469
470    ...
471
472    /**
473     * Exit the class Model.
474     */
475    void exit class Model() {
476    }
477
478    ...
479
479    /**
480     * Exit the class Model.
481     */
482    void exit class Model() {
483    }
484
485    ...
486
486    /**
487     * Exit the class Model.
488     */
489    void exit class Model() {
490    }
491
492    ...
493
493    /**
494     * Exit the class Model.
495     */
496    void exit class Model() {
497    }
498
499    ...
499
499    /**
500     * Exit the class Model.
501     */
502    void exit class Model() {
503    }
504
505    ...
506
506    /**
507     * Exit the class Model.
508     */
509    void exit class Model() {
510    }
511
512    ...
513
513    /**
514     * Exit the class Model.
515     */
516    void exit class Model() {
517    }
518
519    ...
519
519    /**
520     * Exit the class Model.
521     */
522    void exit class Model() {
523    }
524
525    ...
525
525    /**
526     * Exit the class Model.
527     */
528    void exit class Model() {
529    }
530
531    ...
531
531    /**
532     * Exit the class Model.
533     */
534    void exit class Model() {
535    }
536
537    ...
537
537    /**
538     * Exit the class Model.
539     */
540    void exit class Model() {
541    }
542
543    ...
543
543    /**
544     * Exit the class Model.
545     */
546    void exit class Model() {
547    }
548
549    ...
549
549    /**
550     * Exit the class Model.
551     */
552    void exit class Model() {
553    }
554
555    ...
555
555    /**
556     * Exit the class Model.
557     */
558    void exit class Model() {
559    }
560
561    ...
561
561    /**
562     * Exit the class Model.
563     */
564    void exit class Model() {
565    }
566
567    ...
567
567    /**
568     * Exit the class Model.
569     */
570    void exit class Model() {
571    }
572
573    ...
573
573    /**
574     * Exit the class Model.
575     */
576    void exit class Model() {
577    }
578
579    ...
579
579    /**
580     * Exit the class Model.
581     */
582    void exit class Model() {
583    }
584
585    ...
585
585    /**
586     * Exit the class Model.
587     */
588    void exit class Model() {
589    }
590
591    ...
591
591    /**
592     * Exit the class Model.
593     */
594    void exit class Model() {
595    }
596
597    ...
597
597    /**
598     * Exit the class Model.
599     */
600    void exit class Model() {
601    }
602
603    ...
603
603    /**
604     * Exit the class Model.
605     */
606    void exit class Model() {
607    }
608
609    ...
609
609    /**
610     * Exit the class Model.
611     */
612    void exit class Model() {
613    }
614
615    ...
615
615    /**
616     * Exit the class Model.
617     */
618    void exit class Model() {
619    }
620
621    ...
621
621    /**
622     * Exit the class Model.
623     */
624    void exit class Model() {
625    }
626
627    ...
627
627    /**
628     * Exit the class Model.
629     */
630    void exit class Model() {
631    }
632
633    ...
633
633    /**
634     * Exit the class Model.
635     */
636    void exit class Model() {
637    }
638
639    ...
639
639    /**
640     * Exit the class Model.
641     */
642    void exit class Model() {
643    }
644
645    ...
645
645    /**
646     * Exit the class Model.
647     */
648    void exit class Model() {
649    }
650
651    ...
651
651    /**
652     * Exit the class Model.
653     */
654    void exit class Model() {
655    }
656
657    ...
657
657    /**
658     * Exit the class Model.
659     */
660    void exit class Model() {
661    }
662
663    ...
663
663    /**
664     * Exit the class Model.
665     */
666    void exit class Model() {
667    }
668
669    ...
669
669    /**
670     * Exit the class Model.
671     */
672    void exit class Model() {
673    }
674
675    ...
675
675    /**
676     * Exit the class Model.
677     */
678    void exit class Model() {
679    }
680
681    ...
681
681    /**
682     * Exit the class Model.
683     */
684    void exit class Model() {
685    }
686
687    ...
687
687    /**
688     * Exit the class Model.
689     */
690    void exit class Model() {
691    }
692
693    ...
693
693    /**
694     * Exit the class Model.
695     */
696    void exit class Model() {
697    }
698
699    ...
699
699    /**
700     * Exit the class Model.
701     */
702    void exit class Model() {
703    }
704
705    ...
705
705    /**
706     * Exit the class Model.
707     */
708    void exit class Model() {
709    }
710
711    ...
711
711    /**
712     * Exit the class Model.
713     */
714    void exit class Model() {
715    }
716
717    ...
717
717    /**
718     * Exit the class Model.
719     */
720    void exit class Model() {
721    }
722
723    ...
723
723    /**
724     * Exit the class Model.
725     */
726    void exit class Model() {
727    }
728
729    ...
729
729    /**
730     * Exit the class Model.
731     */
732    void exit class Model() {
733    }
734
735    ...
735
735    /**
736     * Exit the class Model.
737     */
738    void exit class Model() {
739    }
740
741    ...
741
741    /**
742     * Exit the class Model.
743     */
744    void exit class Model() {
745    }
746
747    ...
747
747    /**
748     * Exit the class Model.
749     */
750    void exit class Model() {
751    }
752
753    ...
753
753    /**
754     * Exit the class Model.
755     */
756    void exit class Model() {
757    }
758
759    ...
759
759    /**
760     * Exit the class Model.
761     */
762    void exit class Model() {
763    }
764
765    ...
765
765    /**
766     * Exit the class Model.
767     */
768    void exit class Model() {
769    }
770
771    ...
771
771    /**
772     * Exit the class Model.
773     */
774    void exit class Model() {
775    }
776
777    ...
777
777    /**
778     * Exit the class Model.
779     */
780    void exit class Model() {
781    }
782
783    ...
783
783    /**
784     * Exit the class Model.
785     */
786    void exit class Model() {
787    }
788
789    ...
789
789    /**
790     * Exit the class Model.
791     */
792    void exit class Model() {
793    }
794
795    ...
795
795    /**
796     * Exit the class Model.
797     */
798    void exit class Model() {
799    }
800
801    ...
801
801    /**
802     * Exit the class Model.
803     */
804    void exit class Model() {
805    }
806
807    ...
807
807    /**
808     * Exit the class Model.
809     */
810    void exit class Model() {
811    }
812
813    ...
813
813    /**
814     * Exit the class Model.
815     */
816    void exit class Model() {
817    }
818
819    ...
819
819    /**
820     * Exit the class Model.
821     */
822    void exit class Model() {
823    }
824
825    ...
825
825    /**
826     * Exit the class Model.
827     */
828    void exit class Model() {
829    }
830
831    ...
831
831    /**
832     * Exit the class Model.
833     */
834    void exit class Model() {
835    }
836
837    ...
837
837    /**
838     * Exit the class Model.
839     */
840    void exit class Model() {
841    }
842
843    ...
843
843    /**
844     * Exit the class Model.
845     */
846    void exit class Model() {
847    }
848
849    ...
849
849    /**
850     * Exit the class Model.
851     */
852    void exit class Model() {
853    }
854
855    ...
855
855    /**
856     * Exit the class Model.
857     */
858    void exit class Model() {
859    }
860
861    ...
861
861    /**
862     * Exit the class Model.
863     */
864    void exit class Model() {
865    }
866
867    ...
867
867    /**
868     * Exit the class Model.
869     */
870    void exit class Model() {
871    }
872
873    ...
873
873    /**
874     * Exit the class Model.
875     */
876    void exit class Model() {
877    }
878
879    ...
879
879    /**
880     * Exit the class Model.
881     */
882    void exit class Model() {
883    }
884
885    ...
885
885    /**
886     * Exit the class Model.
887     */
888    void exit class Model() {
889    }
890
891    ...
891
891    /**
892     * Exit the class Model.
893     */
894    void exit class Model() {
895    }
896
897    ...
897
897    /**
898     * Exit the class Model.
899     */
900    void exit class Model() {
901    }
902
903    ...
903
903    /**
904     * Exit the class Model.
905     */
906    void exit class Model() {
907    }
908
909    ...
909
909    /**
910     * Exit the class Model.
911     */
912    void exit class Model() {
913    }
914
915    ...
915
915    /**
916     * Exit the class Model.
917     */
918    void exit class Model() {
919    }
920
921    ...
921
921    /**
922     * Exit the class Model.
923     */
924    void exit class Model() {
925    }
926
927    ...
927
927    /**
928     * Exit the class Model.
929     */
930    void exit class Model() {
931    }
932
933    ...
933
933    /**
934     * Exit the class Model.
935     */
936    void exit class Model() {
937    }
938
939    ...
939
939    /**
940     * Exit the class Model.
941     */
942    void exit class Model() {
943    }
944
945    ...
945
945    /**
946     * Exit the class Model.
947     */
948    void exit class Model() {
949    }
950
951    ...
951
951    /**
952     * Exit the class Model.
953     */
954    void exit class Model() {
955    }
956
957    ...
957
957    /**
958     * Exit the class Model.
959     */
960    void exit class Model() {
961    }
962
963    ...
963
963    /**
964     * Exit the class Model.
965     */
966    void exit class Model() {
967    }
968
969    ...
969
969    /**
970     * Exit the class Model.
971     */
972    void exit class Model() {
973    }
974
975    ...
975
975    /**
976     * Exit the class Model.
977     */
978    void exit class Model() {
979    }
980
981    ...
981
981    /**
982     * Exit the class Model.
983     */
984    void exit class Model() {
985    }
986
987    ...
987
987    /**
988     * Exit the class Model.
989     */
990    void exit class Model() {
991    }
992
993    ...
993
993    /**
994     * Exit the class Model.
995     */
996    void exit class Model() {
997    }
998
999    ...
999
999    /**
1000    * Exit the class Model.
1001   */
1002   void exit class Model() {
1003   }
1004
1005   ...
1006
1006   /**
1007   * Exit the class Model.
1008   */
1009   void exit class Model() {
1010  }
1011
1012   ...
1013
1013   /**
1014   * Exit the class Model.
1015   */
1016   void exit class Model() {
1017  }
1018
1019   ...
1020
1020   /**
1021   * Exit the class Model.
1022   */
1023   void exit class Model() {
1024  }
1025
1026   ...
1026
1026   /**
1027   * Exit the class Model.
1028   */
1029   void exit class Model() {
1030  }
1031
1032   ...
1032
1032   /**
1033   * Exit the class Model.
1034   */
1035   void exit class Model() {
1036  }
1037
1038   ...
1038
1038   /**
1039   * Exit the class Model.
1040   */
1041   void exit class Model() {
1042  }
1043
1044   ...
1044
1044   /**
1045   * Exit the class Model.
1046   */
1047   void exit class Model() {
1048  }
1049
1050   ...
1050
1050   /**
1051   * Exit the class Model.
1052   */
1053   void exit class Model() {
1054  }
1055
1056   ...
1056
1056   /**
1057   * Exit the class Model.
1058   */
1059   void exit class Model() {
1060  }
1061
1062   ...
1062
1062   /**
1063   * Exit the class Model.
1064   */
1065   void exit class Model() {
1066  }
1067
1068   ...
1068
1068   /**
1069   * Exit the class Model.
1070   */
1071   void exit class Model() {
1072  }
1073
1074   ...
1074
1074   /**
1075   * Exit the class Model.
1076   */
1077   void exit class Model() {
1078  }
1079
1080   ...
1080
1080   /**
1081   * Exit the class Model.
1082   */
1083   void exit class Model() {
1084  }
1085
1086   ...
1086
1086   /**
1087   * Exit the class Model.
1088   */
1089   void exit class Model() {
1090  }
1091
1092   ...
1092
1092   /**
1093   * Exit the class Model.
1094   */
1095   void exit class Model() {
1096  }
1097
1098   ...
1098
1098   /**
1099   * Exit the class Model.
1100  */
1101  void exit class Model() {
1102  }
1103
1104   ...
1105
1105   /**
1106   * Exit the class Model.
1107   */
1108   void exit class Model() {
1109  }
1110
1111   ...
1111
1111   /**
1112   * Exit the class Model.
1113   */
1114   void exit class Model() {
1115  }
1116
1117   ...
1117
1117   /**
1118   * Exit the class Model.
1119   */
1120   void exit class Model() {
1121  }
1122
1123   ...
1123
1123   /**
1124   * Exit the class Model.
1125   */
1126   void exit class Model() {
1127  }
1128
1129   ...
1129
1129   /**
1130   * Exit the class Model.
1131   */
1132   void exit class Model() {
1133  }
1134
1135   ...
1135
1135   /**
1136   * Exit the class Model.
1137   */
1138   void exit class Model() {
1139  }
1140
1141   ...
1141
1141   /**
1142   * Exit the class Model.
1143   */
1144   void exit class Model() {
1145  }
1146
1147   ...
1147
1147   /**
1148   * Exit the class Model.
1149   */
1150   void exit class Model() {
1151  }
1152
1153   ...
1153
1153   /**
1154   * Exit the class Model.
1155   */
1156   void exit class Model() {
1157  }
1158
1159   ...
1159
1159   /**
1160   * Exit the class Model.
1161   */
1162   void exit class Model() {
1163  }
1164
1165   ...
1165
1165   /**
1166   * Exit the class Model.
1167   */
1168   void exit class Model() {
1169  }
1170
1171   ...
1171
1171   /**
1172   * Exit the class Model.
1173   */
1174   void exit class Model() {
1175  }
1176
1177   ...
1177
1177   /**
1178   * Exit the class Model.
1179   */
1180   void exit class Model() {
1181  }
1182
1183   ...
1183
1183   /**
1184   * Exit the class Model.
1185   */
1186   void exit class Model() {
1187  }
1188
1189   ...
1189
1189   /**
1190   * Exit the class Model.
1191   */
1192   void exit class Model() {
1193  }
1194
1195   ...
1195
1195   /**
1196   * Exit the class Model.
1197   */
1198   void exit class Model() {
1199  }
1200
1201   ...
1201
1201   /**
1202   * Exit the class Model.
1203   */
1204   void exit class Model() {
1205  }
1206
1207   ...
1207
1207   /**
1208   * Exit the class Model.
1209   */
1210   void exit class Model() {
1211  }
1212
1213   ...
1213
1213   /**
1214   * Exit the class Model.
1215   */
1216   void exit class Model() {
1217  }
1218
1219   ...
1219
1219   /**
1220   * Exit the class Model.
1221   */
1222   void exit class Model() {
1223  }
1224
1225   ...
1225
1225   /**
1226   * Exit the class Model.
1227   */
1228   void exit class Model() {
1229  }
1230
1231   ...
1231
1231   /**
1232   * Exit the class Model.
1233   */
1234   void exit class Model() {
1235  }
1236
1237   ...
1237
1237   /**
1238   * Exit the class Model.
1239   */
1240   void exit class Model() {
1241  }
1242
1243   ...
1243
1243   /**
1244   * Exit the class Model.
1245   */
1246   void exit class Model() {
1247  }
1248
1249   ...
1249
1249   /**
1250   * Exit the class Model.
1251   */
1252   void exit class Model() {
1253  }
1254
1255   ...
1255
1255   /**
1256   * Exit the class Model.
1257   */
1258   void exit class Model() {
1259  }
1260
1261   ...
1261
1261   /**
1262   * Exit the class Model.
1263   */
1264   void exit class Model() {
1265  }
1266
1267   ...
1267
1267   /**
1268   * Exit the class Model.
1269   */
1270   void exit class Model() {
1271  }
1272
1273   ...
1273
1273   /**
1274   * Exit the class Model.
1275   */
1276   void exit class Model() {
1277  }
1278
1279   ...
1279
1279   /**
1280   * Exit the class Model.
1281   */
1282   void exit class Model() {
1283  }
1284
1285   ...
1285
1285   /**
1286   * Exit the class Model.
1287   */
1288   void exit class Model() {
1289  }
1290
1291   ...
1291
1291   /**
1292   * Exit the class Model.
1293   */
1294   void exit class Model() {
1295  }
1296
1297   ...
1297
1297   /**
1298   * Exit the class Model.
1299   */
1300   void exit class Model() {
1301  }
1302
1303   ...
1303
1303   /**
1304   * Exit the class Model.
1305   */
1306   void exit class Model() {
1307  }
1308
1309   ...
1309
1309   /**
1310   * Exit the class Model.
1311   */
1312   void exit class Model() {
1313  }
1314
1315   ...
1315
1315   /**
1316   * Exit the class Model.
1317   */
1318   void exit class Model() {
1319  }
1320
1321   ...
1321
1321   /**
1322   * Exit the class Model.
1323   */
1324   void exit class Model() {
1325  }
1326
1327   ...
1327
1327   /**
1328   * Exit the class Model.
1329   */
1330   void exit class Model() {
1331  }
1332
1333   ...
1333
1333   /**
1334   * Exit the class Model.
1335   */
1336   void exit class Model() {
1337  }
1338
1339   ...
1339
1339   /**
1340   * Exit the class Model.
1341   */
1342   void exit class Model() {
1343  }
1344
1345   ...
1345
1345   /**
1346   * Exit the class Model.
1347   */
1348   void exit class Model() {
1349  }
1350
1351   ...
1351
1351   /**
1352   * Exit the class Model.
1353   */
1354   void exit class Model() {
1355  }
1356
1357   ...
1357
1357   /**
1358   * Exit the class Model.
1359   */
1360   void exit class Model() {
1361  }
1362
1363   ...
1363
1363   /**
1364   * Exit the class Model.
1365   */
1366   void exit class Model() {
1367  }
1368
1369   ...
1369
1369   /**
1370   * Exit the class Model.
1371   */
1372   void exit class Model() {
1373  }
1374
1375   ...
1375
1375   /**
1376   * Exit the class Model.
1377   */
1378   void exit class Model() {
1379  }
1380
1381   ...
1381
1381   /**
1382   * Exit the class Model.
1383   */
1384   void exit class Model() {
1385  }
1386
1387   ...
1387
1387   /**
1388   * Exit the class Model.
1389   */
1390   void exit class Model() {
1391  }
1392
1393   ...
1393
1393   /**
1394   * Exit the class Model.
1395   */
1396   void exit class Model() {
1397  }
1398
1399   ...
1399
1399   /**
1400   * Exit the class Model.
1401   */
1402   void exit class Model() {
1403  }
1404
1405   ...
1405
1405   /**
1406   * Exit the class Model.
1407   */
1408   void exit class Model() {
1409  }
1410
1411   ...
1411
1411   /**
1412   * Exit the class Model.
1413   */
1414   void exit class Model() {
1415  }
1416
1417   ...
1417
1417   /**
1418   * Exit the class Model.
1419   */
1420   void exit class Model() {
1421  }
1422
1423   ...
1423
1423   /**
1424   * Exit the class Model.
1425   */
1426   void exit class Model() {
1427  }
1428
1429   ...
1429
1429   /**
1430   * Exit the class Model.
1431   */
1432   void exit class Model() {
1433  }
1434
1435   ...
1435
1435   /**
1436   * Exit the class Model.
1437   */
1438   void exit class Model() {
1439  }
1440
1441   ...
1441
1441   /**
1442   * Exit the class Model.
1443   */
1444   void exit class Model() {
1445  }
1446
1447   ...
1447
1447   /**
1448   * Exit the class Model.
1449   */
1450   void exit class Model() {
1451  }
1452
1453   ...
1453
1453   /**
1454   * Exit the class Model.
1455   */
1456   void exit class Model() {
1457  }
1458
1459   ...
1459
1459   /**
1460   * Exit the class Model.
1461   */
1462   void exit class Model() {
1463  }
1464
1465   ...
1465
1465   /**
1466   * Exit the class Model.
1467   */
1468   void exit class Model() {
1469  }
1470
1471   ...
1471
1471   /**
1472   * Exit the class Model.
1473   */
1474   void exit class Model() {
1475  }
1476
1477   ...
1477
1477   /**
1478   * Exit the class Model.
1479   */
1480   void exit class Model() {
1481  }
1482
1483   ...
1483
1483   /**
1484   * Exit the class Model.
1485   */
1486   void exit class Model() {
1487  }
1488
1489   ...
1489
1489   /**
1490   * Exit the class Model.
1491   */
1492   void exit class Model() {
1493  }
1494
1495   ...
1495
1495   /**
1496   * Exit the class Model.
1497   */
1498   void exit class Model() {
1499  }
1500
1501   ...
1501
1501   /**
1502   * Exit the class Model.
1503   */
1504   void exit class Model() {
1505  }
1506
1507   ...
1507
1507   /**
1508   * Exit the class Model.
1509   */
1510   void exit class Model() {
1511  }
1512
1513   ...
1513
1513   /**
1514   * Exit the class Model.
1515   */
1516   void exit class Model() {
1517  }
1518
1519   ...
1519
1519   /**
1520   * Exit the class Model.
1521   */
1522   void exit class Model() {
1523  }
1524
1525   ...
1525
1525   /**
1526   * Exit the class Model.
1527   */
1528   void exit class Model() {
1529  }
1530
1531   ...
1531
1531   /**
1532   * Exit the class Model.
1533   */
1534   void exit class Model() {
1535  }
1536
1537   ...
1537
1537   /**
1538   * Exit the class Model.
1539   */
1540   void exit class Model() {
1541  }
1542
1543   ...
1543
```

Control flow graph for a class

Control flow for each method

+
node for class

+
edges

from node *class* to the start
nodes of the methods
from the end nodes of the
methods to node *class*

=> control flow through
sequences of method calls

