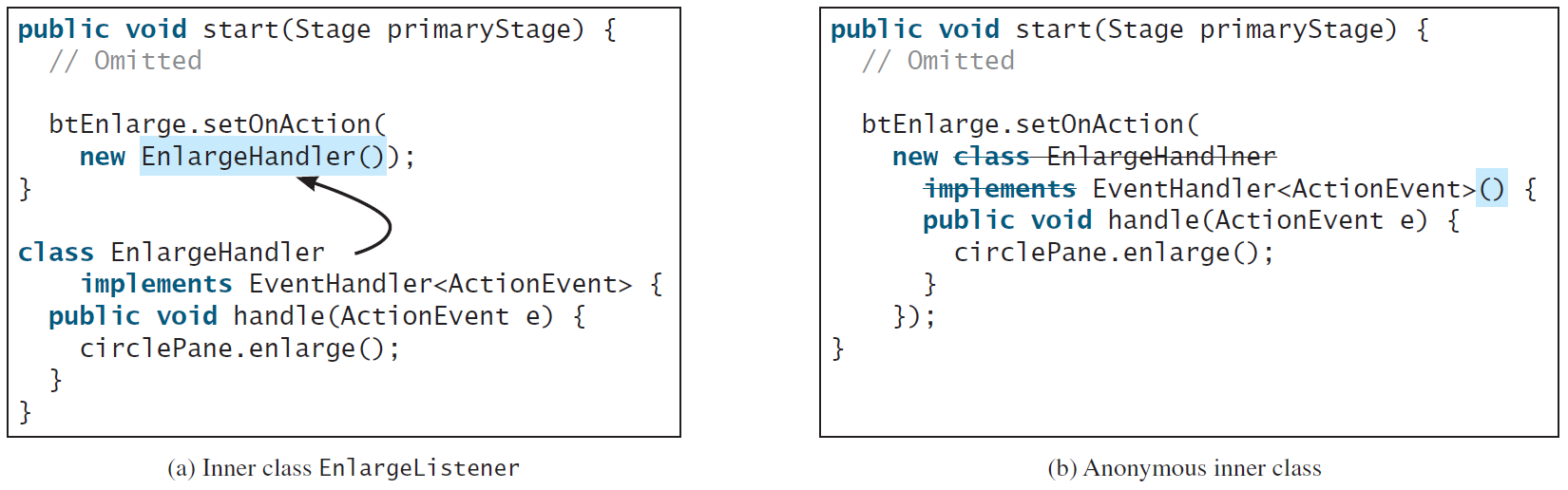
CS 150 Topics List

Chapter 15

1. Events (15.1)
   1. Source object
   2. Event object
   3. Handler object
      1. Must implement EventHandler<T extends Event> which contains handle(ActionEVent)
      2. Must be registered with the event source object using source.setOnAction(handler);
2. User actions listed with source objects, event types fired, and registration methods (p 589).
3. Registering Handlers
   1. The handler object must be an instance of the corresponding event-handler interface.
4. Inner Classes
   1. Also called nested classes.
   2. An inner class is a member of another class.
   3. Advantages: In some applications, you can use an inner class to make programs simple.
   4. An inner class can reference the data and methods defined in the outer class in which it nests, so you do not need to pass the reference of the outer class to the constructor of the inner class.
   5. Are compiled into a classes named *OuterClassName*$*InnerClassName*.class.
   6. An inner class can be declared public, protected, or private subject to the same visibility rules applied to a member of the class.
   7. An inner class can be declared static. A static inner class can be accessed using the outer class name. A static inner class cannot access nonstatic members of the outer class.
5. Anonymous Inner Classes
   1. An anonymous inner class must always extend a superclass or implement an interface, but it cannot have an explicit extends or implements clause.
   2. An anonymous inner class must implement all the abstract methods in the superclass or in the interface.
   3. An anonymous inner class always uses the no-arg constructor from its superclass to create an instance. If an anonymous inner class implements an interface, the constructor is Object().
   4. An anonymous inner class is compiled into a class named OuterClassName$*n*.class. For example, if the outer class Test has two anonymous inner classes, these two classes are compiled into Test$1.class and Test$2.class.
   5. Inner class listeners can be shortened using anonymous inner classes. An *anonymous inner class* is an inner class without a name. It combines declaring an inner class and creating an instance of the class in one step.
6. Lamda expressions. (15.6)
   1. Simplify creating anonymous inner classes.
   2. Require Java 8.
   3. Can only be implemented on interfaces that contain only one abstract method. These interfaces are called functional interfaces and/or Single Abstract Method (SAM) interfaces.



1. Listeners for Observable Objects(15.10)
   1. Instances of Observable are known as Observable objects.
   2. They contain the addListener(InvalidationListener listener) method for adding a listener to a property that will run whenever the property being listened to is changed.
   3. ObservablePropertyDemo and DisplayResizableClock
2. Animation class(15.11)
   1. Allows for use of animations
   2. PathTransition moves a node along a path
   3. PathTransitionDemo
   4. FadeTransitionDemo shows how FadeTransitions work.
   5. Timeline class
      1. Uses KeyFrames to sequence animations
      2. KeyFrames are executed sequentially at intervals that you set up.