**Javafx‌ ‌Event‌ ‌Handling‌ ‌**

 ‌

 ‌

***Figure‌ ‌from‌ ‌textbook‌ ‌page‌ ‌586‌ ‌***

‌

**java.util.EventObject‌**

**‌**

‌***Figure‌ ‌from‌ ‌textbook‌ ‌page‌ ‌588‌ ‌***

***‌***

***‌***

***‌***

***‌***

 ‌

***‌***

***Figure‌ ‌from‌ ‌textbook‌ ‌page‌ ‌589‌ ‌***

***‌***

***Figure‌ ‌from‌ ‌textbook‌ ‌page‌ ‌590‌ ‌***

**Inner‌ ‌Classes‌ ‌**

We‌ ‌use‌ ‌inner‌ ‌classes‌ ‌to‌ ‌create‌ ‌a‌ ‌costume‌ ‌Event‌ ‌handling‌ ‌class‌ ‌(listener)‌ ‌



 ‌

***Example‌ ‌code.‌ ‌***

***-‌ ‌A‌ ‌is‌ ‌compiled‌ ‌into‌ ‌name‌ ‌Test$A.class‌ ‌***

***-‌ ‌A‌ ‌can‌ ‌reference‌ ‌data‌ ‌and‌ ‌methods‌ ‌from‌ ‌Test‌ ‌(e.g.‌ ‌data,‌ ‌m()‌ ‌)‌ ‌***

***-A‌ ‌may‌ ‌use‌ ‌visibility‌ ‌modifiers‌ ‌(public,‌ ‌default,‌ ‌protected,‌ ‌private)‌ ‌***

***-‌ ‌A‌ ‌may‌ ‌be‌ ‌static:‌ ‌***

***-‌ ‌is‌ ‌accessed‌ ‌using‌ ‌Test.A‌ ‌***

***-‌ ‌if‌ ‌static‌ ‌A‌ ‌cannot‌ ‌access‌ ‌non-static‌ ‌members‌ ‌of‌ ‌Test‌ ‌***

***-‌ ‌Creating‌ ‌objects‌ ‌of‌ ‌non-static‌ ‌A‌ ‌in‌ ‌other‌ ‌classes:‌ ‌***

***Test‌ ‌t‌ ‌=‌ ‌new‌ ‌Test();‌ ‌then‌ ‌Test.A‌ ‌a‌ ‌=‌ ‌t.new‌ ‌A();‌ ‌***

 ***‌for‌ ‌static‌ ‌A:‌ ‌***

***Test.A‌ ‌a‌ ‌=‌ ‌new‌ ‌Test.A();‌ ‌***

‌

 ‌

***‌***

**Anonymous‌ ‌Inner‌ ‌Classes‌ ‌**

**inner‌ ‌classes‌ ‌without‌ ‌a‌ ‌name.‌ ‌**

**structure:‌ ‌**

**new‌ ‌Superclass/Interface(){‌ ‌**

**}‌ ‌**

**e.g.‌ ‌new‌ ‌EvenHandler<ActionEvent>(){‌ ‌**

**}‌ ‌**

**-‌ ‌must‌ ‌always‌ ‌extend‌ ‌a‌ ‌superclass‌ ‌or‌ ‌implement‌ ‌an‌ ‌interface‌ ‌(‌ ‌Without‌ ‌using‌ ‌the‌ ‌words‌ ‌**

**extends‌ ‌or‌ ‌implements)‌ ‌**

**-‌ ‌Must‌ ‌implement‌ ‌all‌ ‌abstract‌ ‌methods‌ ‌in‌ ‌superclass/interface‌ ‌**

**-‌ ‌Uses‌ ‌the‌ ‌no-arg‌ ‌constructor‌ ‌from‌ ‌superclass‌ ‌to‌ ‌create‌ ‌an‌ ‌instance‌ ‌**

**-‌ ‌Uses‌ ‌constructor‌ ‌Object()‌ ‌if‌ ‌it‌ ‌implements‌ ‌an‌ ‌interface‌ ‌**

**-‌ ‌Compiled‌ ‌into‌ ‌Test$1.class,‌ ‌Test$2.class,‌ ‌…‌ ‌**

**‌**

**Lambda‌ ‌Expressions‌ ‌**

**(type‌ ‌x,‌ ‌type,‌ ‌y)‌ ‌->‌ ‌statement‌ ‌(No‌ ‌semicolon)‌ ‌**

(type‌ ‌x,‌ ‌type‌ ‌y)‌ ‌->{statement;‌ ‌statement;…;}‌ ‌

-‌ ‌‌***Functional‌ ‌Interface‌ ‌(‌ ‌Single‌ ‌Abstract‌ ‌Method‌ ‌(SAM)‌ ‌interface)‌*** ‌

-‌ ‌necessary‌ ‌for‌ ‌compilers‌ ‌to‌ ‌understand‌ ‌lambda‌ ‌expressions‌ ‌

 ‌

***‌***