Sabancı Üniversitesi

Inner Classes

Sabancı Üniversitesi

What is an Inner Class?

- A class that is defined within another class
- Benefits:
  - can access private data of the outer class
  - Anonymous inner classes can be defined inline in your code
  - can be hidden from the outer classes within the package
  - useful for adding event listener capabilities
- There are three types of inner classes
  - member - defined within the outer class (either static or non-static)
  - Local - defined within a method
  - Anonymous - defined within a method, has no name and can implement an interface

Sabancı Üniversitesi

Notes About Top Level Classes

- All code in Java is implemented within a class
- A Java source file must be named ClassName.java
  - The source file may contain many class definitions, but there can only be one public class in a source file
- Top level class definitions can only be defined with `public` and/or default accessor (but sure can be `final` or `abstract`)

Sabancı Üniversitesi

Member classes (non-static)

- can access all fields/methods of enclosing class
- may not contain static member (unless declared as final)
- may not contain static initializers

```
public class Book {
  private String title;
  private List<Chapter> chapters = new ArrayList<Chapter>();

  class Chapter{
    public String toString(){
      int totalChapters = chapters.size(); //from enclosing class
      return String.format("Chapter '%s' (%d of %d)", title,chapterNumber,
        totalChapters);
    }
    private String title;
    private int chapterNumber;
    ...
  }
  ...
}
```

Sabancı Üniversitesi

Member classes (non-static)

- can be instantiated using reference to outer class

```
Book.Chapter chapter1 = new Book().new Chapter("chapter 1",1);

Book book = new Book();
Book.Chapter chapter2 = book.new Chapter("chapter 1",1);
```

- within enclosing class inner class can be used as any other class

```
Chapter c = new Chapter("chapter 1",1);
```

Sabancı Üniversitesi

Member Classes/Interfaces (static)

- Defined as static member of enclosing class
- can only access static fields/methods of the enclosing class

```
public class Book {
  private Publisher publisher;

  static class Publisher{
    String name;
    String country;
  }
  ...
}
```

- can be constructed without an instance of enclosing class

```
Book.Publisher publisher = new Book.Publisher();
```

Sabancı Üniversitesi

Member Classes/Interfaces (static)

- Behaves like a normal static member of enclosing class

```
Book.Publisher publisher = new Book.Publisher();
publisher.name = "Fred";

Book book1 = new Book();
book1.setPublisher(publisher);
Book book2 = new Book();
book2.setPublisher(publisher);

publisher.name = "Jen";

System.out.println(book1.getPublisher().name); //shows 'Jen'
System.out.println(book2.getPublisher().name); //shows 'Jen'
```

Sabancı Üniversitesi

Local Inner Classes

- Defined within a method
- cannot be declared static
- cannot have an access modifier
- can access enclosing class's methods and fields
- can only access final variables of the enclosing method

```
public void addCover(){

  class CoverPage implements Page{
    boolean inColor;
    public String getText(){
      return "Title: " + title; //property of enclosing class
    }

    CoverPage(boolean inColor){
      this.inColor = inColor;
    }
  }

  Page colorCover = new CoverPage(true);
  Page bwCover = new CoverPage(false);
  ...
}
```

Sabancı Üniversitesi

Anonymous Classes

- Declared without a name
- allows objects to be created on the fly
- may not define extends or implements
- may not contain constructors

```
public void addCommentPage(final String text){
  addPage(new Page(){
    public String getText(){
      return "Comment: " + text;
    }
  });
}

public void addPage(Page p){
  ...
}
```

Sabancı Üniversitesi

Anonymous Classes

- Anonymous classes can directly implement an interface or extend from a class
- Implementing an interface:

```
interface Page{
  public String getText();
}

addPage(new Page(){
  public String getText(){
    return "Comment: " + text;
  }
});
```

Sabancı Üniversitesi

Anonymous Classes

- Extending a class:

```
class Chapter{
  public String toString(){
    int totalChapters = chapters.size(); //from enclosing class
    return String.format("Chapter '%s' (%d of %d)", title,chapterNumber,
      totalChapters);
  }
  private String title;
  private int chapterNumber;

  public Chapter(String title, int chapterNumber){
    this.title = title;
    this.chapterNumber = chapterNumber;
  }
}
```

class to be extended

```
public void addIntroChapter(){
  addChapter(new Chapter("Intro",0){
    public String toString(){
      return String.format("Chapter '%s' ", title);
    }
  });
}

public void addChapter(Chapter c){
  this.chapters.add(c);
}
```

extending the class in method