

# OOP Revisited

Examining a Song and CD class

# Song Class

- Needs to represent all the relevant information about a song
  - title
  - composer
- Needs to support actions that you can perform on a song
  - getters/setters
  - commands/questions

# Example Song Class

```
public class Song {  
    private String name, artist, album;  
    private int trackNumber;  
    private int lengthInSeconds;  
  
    public Song(String n, String ar, String al, int tn, int lis) {  
        name = n;  
        artist = ar;  
        album = al;  
        trackNumber = tn;  
        lengthInSeconds = lis;  
    }  
}
```

# Example Song Class

```
public class Song {  
    private String name, artist, album;  
    private int trackNumber;  
    private int lengthInSeconds;
```

private  
instance fields

```
    public Song(String n, String ar, String al, int tn, int lis) {  
        name = n;  
        artist = ar;  
        album = al;  
        trackNumber = tn;  
        lengthInSeconds = lis;  
    }  
}
```

# Example Song Class

```
public class Song {  
    private String name, artist, album;  
    private int trackNumber;  
    private int lengthInSeconds;
```

private  
instance fields

```
    public Song(String n, String ar, String al, int tn, int lis) {  
        name = n;  
        artist = ar;  
        album = al;  
        trackNumber = tn;  
        lengthInSeconds = lis;  
    }  
}
```

parameter list

# Example Song Class

```
public class Song {  
    private String name, artist, album;  
    private int trackNumber;  
    private int lengthInSeconds;
```

private  
instance fields

```
    public Song(String n, String ar, String al, int tn, int lis) {  
        name = n;  
        artist = ar;  
        album = al;  
        trackNumber = tn;  
        lengthInSeconds = lis;  
    }  
}
```

parameter list

constructor/  
class name

# Song Class

```
public Song() {  
    name = "Default";  
    artist = "Default";  
    album = "Default";  
    trackNumber = 1;  
    lengthInSeconds = 300;  
}
```

# Song Class

```
public Song() {  
    name = "Default";  
    artist = "Default";  
    album = "Default";  
    trackNumber = 1;  
    lengthInSeconds = 300;  
}
```

Another  
constructor

# Song Class

```
public String getName() { return name; }  
public String getArtist() { return artist; }  
public String getAlbum() { return album; }  
public int getTrackNumber() { return trackNumber; }  
public int getLengthInSeconds() { return lengthInSeconds; }
```

# Song Class

```
public String getName() { return name; }  
public String getArtist() { return artist; }  
public String getAlbum() { return album; }  
public int getTrackNumber() { return trackNumber; }  
public int getLengthInSeconds() { return lengthInSeconds; }
```

Getter  
Methods

# Using the Song Class

```
Song a = new Song("Just Dance", "Lady Gaga", "The Fame", 1, 242);  
Song b = new Song();
```

```
String bName = b.getName();  
int aLen = a.getLengthInSeconds();
```

Call Song methods off of Song variables.

# CD Class

- Should hold a bunch of songs
- Should support basic operations like picking a new song

# CD Class

```
public class CD {  
    private ArrayList<Song> songs;  
    private int currentSong;  
  
    //creates a CD with one default song  
    public CD() { /*TO FINISH*/ }  
  
    public void addSong(Song s) { /*IMPLEMENTATION NOT SHOW*/ }  
  
    //picks the next song - should loop around  
    public void nextSong() { /*TO FINISH*/ }  
  
    //switches to the first song whose name contains the parameter  
    public void switchToSong(String name) { /*TO FINISH*/ }  
}
```

# CD Constructor

# CD Constructor

//creates a CD with one default song

# CD Constructor

```
//creates a CD with one default song  
public CD() {
```

# CD Constructor

```
//creates a CD with one default song  
public CD() {  
  
    songs = new ArrayList<Song>();
```

# CD Constructor

```
//creates a CD with one default song  
public CD() {  
  
    songs = new ArrayList<Song>();  
    songs.add(new Song());  
}
```

# CD Constructor

```
//creates a CD with one default song  
public CD() {  
  
    songs = new ArrayList<Song>();  
    songs.add(new Song());  
    currentSong = 0;  
}
```

# CD Constructor

```
//creates a CD with one default song
public CD() {

    songs = new ArrayList<Song>();
    songs.add(new Song());
    currentSong = 0;

}
```

# CD Constructor

```
//creates a CD with one default song
public CD() {

    songs = new ArrayList<Song>();
    songs.add(new Song());
    currentSong = 0;

}
```

Purpose of  
constructor:  
Initialize the  
fields!

# Next Song

# Next Song

//picks the next song - should loop around

# Next Song

```
//picks the next song - should loop around  
public void nextSong() {
```

# Next Song

```
//picks the next song - should loop around  
public void nextSong() {
```

```
    currentSong++;
```

# Next Song

```
//picks the next song - should loop around  
public void nextSong() {
```

```
    currentSong++;  
    if(currentSong >= songs.size())
```

# Next Song

```
//picks the next song - should loop around  
public void nextSong() {
```

```
    currentSong++;  
    if(currentSong >= songs.size())  
        currentSong = 0;
```

# Next Song

```
//picks the next song - should loop around  
public void nextSong() {  
  
    currentSong++;  
    if(currentSong >= songs.size())  
        currentSong = 0;  
  
}
```

# Next Song

```
//picks the next song - should loop around  
public void nextSong() {
```

```
    currentSong++;  
    if(currentSong >= songs.size())  
        currentSong = 0;
```

```
}
```

Change the  
instance fields!

# Find Song

## Songs

"Run", "Snow Patrol", "Final Straw", 7, 357

"First snow", "TSO", ...

"Viva la vida", "Coldplay", ...

"Thunderstruck", "ACDC", ...

"Dream on", "Aerosmith", ...

"Your woman", "White Town", ...

"Right now", "Van Halen"

# Find Song

| Songs                                       |
|---|
| "Run", "Snow Patrol", "Final Straw", 7, 357 |
| "First snow", "TSO", ...                    |
| "Viva la vida", "Coldplay", ...             |
| "Thunderstruck", "ACDC", ...                |
| "Dream on", "Aerosmith", ...                |
| "Your woman", "White Town", ...             |
| "Right now", "Van Halen"                    |

switchToSong("Run")  
↓  
song 0

# Find Song

| Songs                                       |
|---|
| "Run", "Snow Patrol", "Final Straw", 7, 357 |
| "First snow", "TSO", ...                    |
| "Viva la vida", "Coldplay", ...             |
| "Thunderstruck", "ACDC", ...                |
| "Dream on", "Aerosmith", ...                |
| "Your woman", "White Town", ...             |
| "Right now", "Van Halen"                    |

switchToSong("Run")

↓  
song 0

switchToSong("run")

↓  
no change

# Find Song

| Songs                                       |
|---|
| "Run", "Snow Patrol", "Final Straw", 7, 357 |
| "First snow", "TSO", ...                    |
| "Viva la vida", "Coldplay", ...             |
| "Thunderstruck", "ACDC", ...                |
| "Dream on", "Aerosmith", ...                |
| "Your woman", "White Town", ...             |
| "Right now", "Van Halen"                    |

switchToSong("Run")

↓  
song 0

switchToSong("run")

↓  
no change

switchToSong("now")

↓  
song 1

# String Reference

```
class java.lang.String
• int length()
• String substring(int from, int to) // returns the substring beginning at from
                                     // and ending at to-1
• String substring(int from)         // returns substring(from, length())
• int indexOf(String str)           // returns the index of the first occurrence of str;
                                     // returns -1 if not found
• int compareTo(String other)       // returns a value < 0 if this is less than other
                                     // returns a value = 0 if this is equal to other
                                     // returns a value > 0 if this is greater than other
```

# Find Song

# Find Song

//switches to the first song that matches the given name

# Find Song

```
//switches to the first song that matches the given name  
public void switchToSong(String name){
```

# Find Song

```
//switches to the first song that matches the given name  
public void switchToSong(String name){  
  
    for(int i = 0; i < songs.size(); i++) {
```

# Find Song

```
//switches to the first song that matches the given name  
public void switchToSong(String name){  
  
    for(int i = 0; i < songs.size(); i++) {  
        Song s = songs.get(i);  
    }  
}
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

```
        if(songName.indexOf(name) != -1) {
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

```
        if(songName.indexOf(name) != -1) {
```

```
            currentSong = i;
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

```
        if(songName.indexOf(name) != -1) {
```

```
            currentSong = i;
```

```
            break;
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

```
        if(songName.indexOf(name) != -1) {
```

```
            currentSong = i;
```

```
            break;
```

```
        }
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

```
        if(songName.indexOf(name) != -1) {
```

```
            currentSong = i;
```

```
            break;
```

```
        }
```

```
    }
```

# Find Song

//switches to the first song that matches the given name

```
public void switchToSong(String name){
```

```
    for(int i = 0; i < songs.size(); i++) {
```

```
        Song s = songs.get(i);
```

```
        String songName = s.getName();
```

```
        if(songName.indexOf(name) != -1) {
```

```
            currentSong = i;
```

```
            break;
```

```
        }
```

```
    }
```

```
}
```

# Oh no!

- The CD player's songs have been jumbled up in the list
- You also want to know which song is the longest

```
//plays the song with the given track number  
public void playTrack(int trackNumber) { /*TO FINISH*/ }
```

```
//returns the longest song - if there's a tie, return the first  
public Song longestSong() { /*TO FINISH */ }
```

# Play Track

```
//plays the song with the given track number
public void playTrack(int trackNumber) {

    for(int i = 0; i < songs.size(); i++) {

        Song s = songs.get(i);
        int tNum = s.getTrackNumber();
        if(tNum == trackNumber) {
            currentSong = i;
            break;
        }

    }

}
```

# Longest Song

# Longest Song

//returns the longest song - if there's a tie, return the first

# Longest Song

```
//returns the longest song - if there's a tie, return the first  
public Song longestSong() {
```

# Longest Song

```
//returns the longest song - if there's a tie, return the first  
public Song longestSong() {  
    Song max = songs.get(0);
```

# Longest Song

```
//returns the longest song - if there's a tie, return the first
public Song longestSong() {
    Song max = songs.get(0);

    for(Song s: songs)
```

# Longest Song

```
//returns the longest song - if there's a tie, return the first
public Song longestSong() {
    Song max = songs.get(0);

    for(Song s: songs)
        if(s.getLengthInSeconds() > max.getLengthInSeconds())
```

# Longest Song

```
//returns the longest song - if there's a tie, return the first
public Song longestSong() {
    Song max = songs.get(0);

    for(Song s: songs)
        if(s.getLengthInSeconds() > max.getLengthInSeconds())
            max = s;
```

# Longest Song

```
//returns the longest song - if there's a tie, return the first
public Song longestSong() {
    Song max = songs.get(0);

    for(Song s: songs)
        if(s.getLengthInSeconds() > max.getLengthInSeconds())
            max = s;

    return max;
}
```

# Longest Song

```
//returns the longest song - if there's a tie, return the first
public Song longestSong() {
    Song max = songs.get(0);

    for(Song s: songs)
        if(s.getLengthInSeconds() > max.getLengthInSeconds())
            max = s;

    return max;
}
```

# Overloaded Methods

- Methods with the same name and return type, but either...
  - different number of parameters
  - different types of parameters (including order)
  - both
- Does NOT include different parameter names

# Add Overloaded

# Add Overloaded

```
public void add(Song s) {
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {  
    if(i >= 0 && i <= songs.size())
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {  
    if(i >= 0 && i <= songs.size())  
        songs.add(i, s);  
}
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {  
    if(i >= 0 && i <= songs.size())  
        songs.add(i, s);  
}
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {  
    if(i >= 0 && i <= songs.size())  
        songs.add(i, s);  
}
```

```
CD cd = new CD();
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {  
    if(i >= 0 && i <= songs.size())  
        songs.add(i, s);  
}
```

```
CD cd = new CD();  
cd.add(new Song());
```

# Add Overloaded

```
public void add(Song s) {  
    songs.add(s);  
}
```

```
public void add(int i, Song s) {  
    if(i >= 0 && i <= songs.size())  
        songs.add(i, s);  
}
```

```
CD cd = new CD();  
cd.add(new Song());  
cd.add(0, new Song());
```

# Pre/Post Conditions

- Preconditions: things you can safely assume are true when writing a method
- Postconditions: things you must guarantee are true after you finish writing a method

# Pre/Post Conditions

- Preconditions: things you can safely assume are true when writing a method
- Postconditions: things you must guarantee are true after you finish writing a method

//precondition: i is a valid index, s is not **null**

# Pre/Post Conditions

- Preconditions: things you can safely assume are true when writing a method
- Postconditions: things you must guarantee are true after you finish writing a method

//precondition: i is a valid index, s is not **null**

//postcondition: s is now in the CD at index i

# Pre/Post Conditions

- Preconditions: things you can safely assume are true when writing a method
- Postconditions: things you must guarantee are true after you finish writing a method

//precondition: i is a valid index, s is not **null**

//postcondition: s is now in the CD at index i

//postcondition: no other songs are moved

# Pre/Post Conditions

- Preconditions: things you can safely assume are true when writing a method
- Postconditions: things you must guarantee are true after you finish writing a method

```
//precondition: i is a valid index, s is not null  
//postcondition: s is now in the CD at index i  
//postcondition: no other songs are moved  
public void replaceSong(int i, Song s){/*TO FINISH*/}
```

# Replace Song

# Replace Song

```
public void replaceSong(int i, Song s){
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

```
public void replaceSong(int i, Song s){
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
    }  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
        songs.add(s);  
    }  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
        songs.add(s);  
    }  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
        songs.add(s);  
    }  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

Good!

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
        songs.add(s);  
    }  
}
```

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

Good!

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
        songs.add(s);  
    }  
}
```

Ignores precondition

# Replace Song

```
public void replaceSong(int i, Song s){  
    songs.set(i, s);  
}
```

Good!

```
public void replaceSong(int i, Song s){  
    if(i >= 0 && i < songs.size()) {  
        songs.remove(i);  
        songs.add(s);  
    }  
}
```

Ignores precondition

Breaks postconditions