CST 3613 Java Application Development
Fall 2022

Mitigation Exam

Comprehensive OOP Exercise

Professor HG Locklear

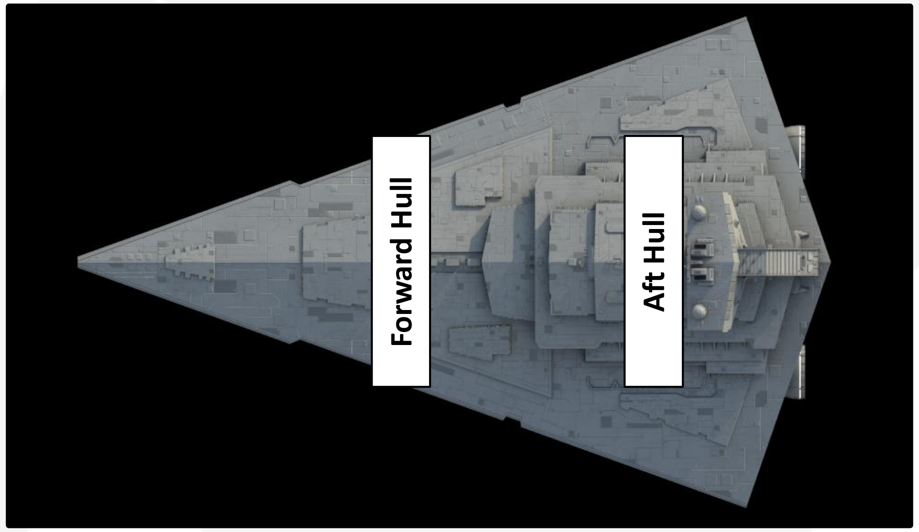
Instructions and Requirements

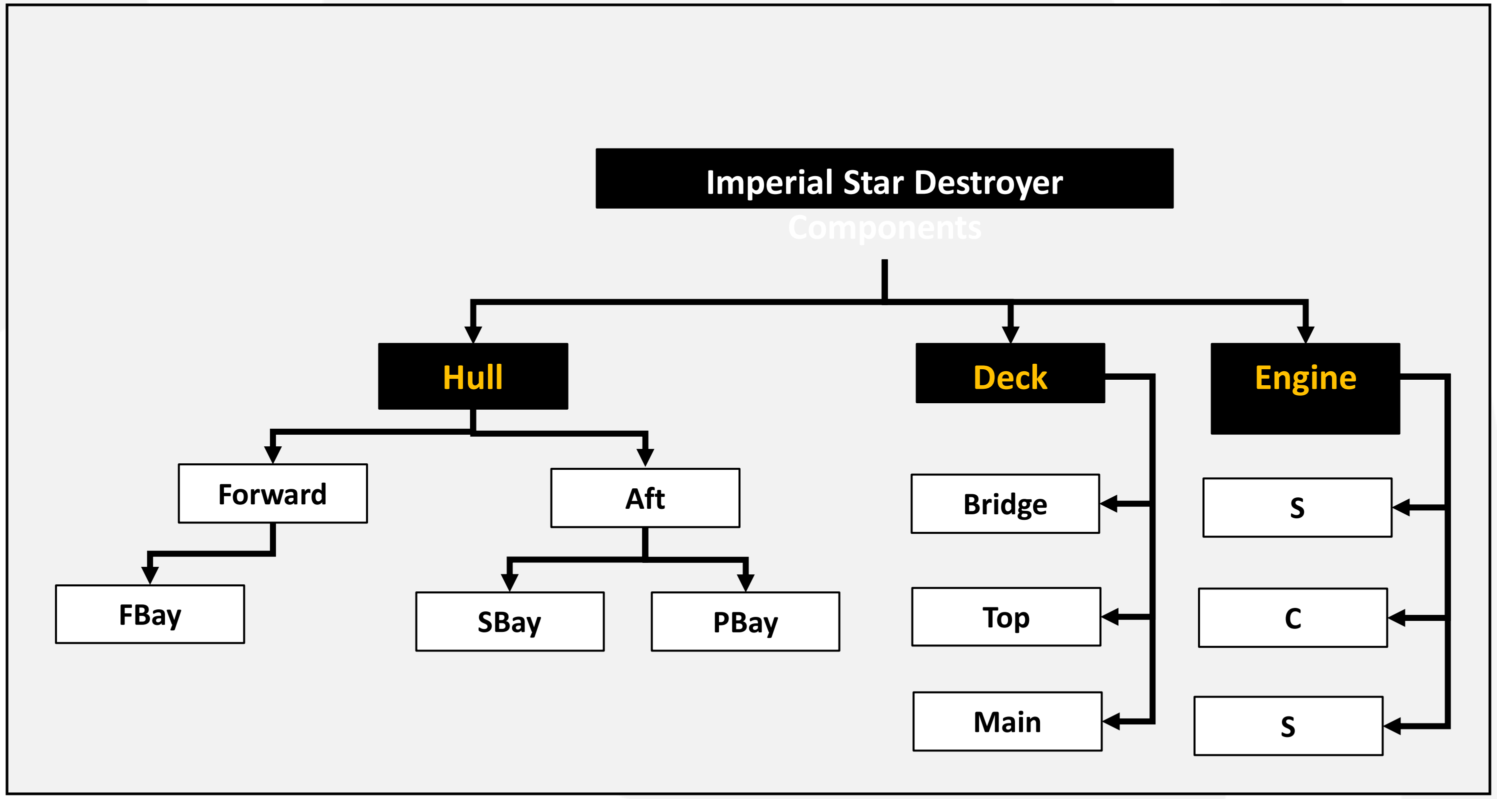
- This is a COMPREHENSIVE OOP Exercise and is designed to demonstrate your understanding of OOP concepts.
- In the completion of this Exercise, you must use the OOP tools you have learned to create a viable solution to this problem. You are given some leeway in decision making…use your understanding to overcome the difficult aspects and remember **YOU MAY NOT ALTER** the Class and Method Specifications as shown in the UML diagrams.
- The SUCCESSFUL completion of this Exercise will help validate your understanding of these concepts and your readiness for the Final Exam.
- This is an OOP problem which involves extensive use of patterns…remember to exploit them to save yourself substantial time in completing the Exercise.
- Any methods whose specification are not given, allow you to implement them in the manner of your choice. Your implementation should be an efficient one and you **must not** hardcode any values in methods which require the displaying of information.
- I will evaluate the solution to this Exercise on the completeness, readability, and efficiency of your solution.
- To receive the maximum points, for this Exercise, requires a well-defined program whose output demonstrates the completeness of your solution .
- Ask any question about the Exercise and be sure to give yourself sufficient time to complete it.

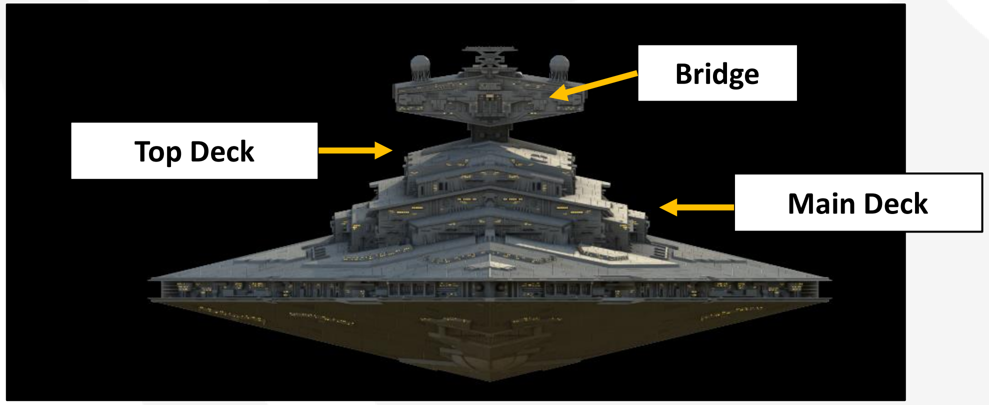
Imperial Star Destroyer

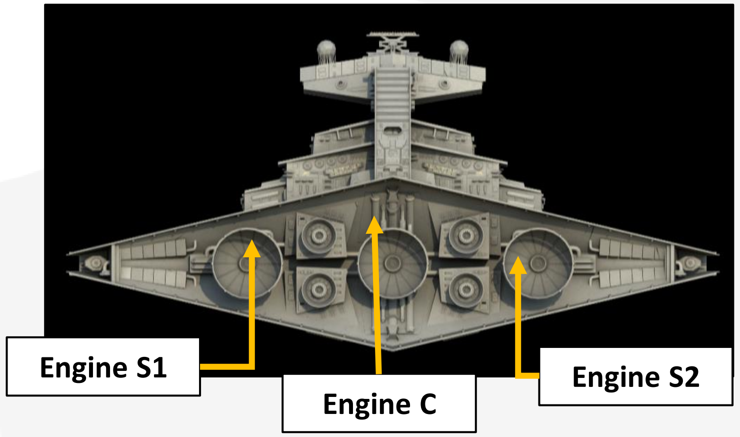
- The **Imperial Star Destroyer** is the main warship of the Galactic Empire.
- There are multiple different variants of the Imperial Star Destroyer.
- **Our task is to create an abstraction of two of the variants of the Imperial Star Destroyer.**
- In order to accomplish this task, we will consider that the construction of an Imperial Star Destroyer **consist of assembling multiple components.**

Imperial Star Destroyer

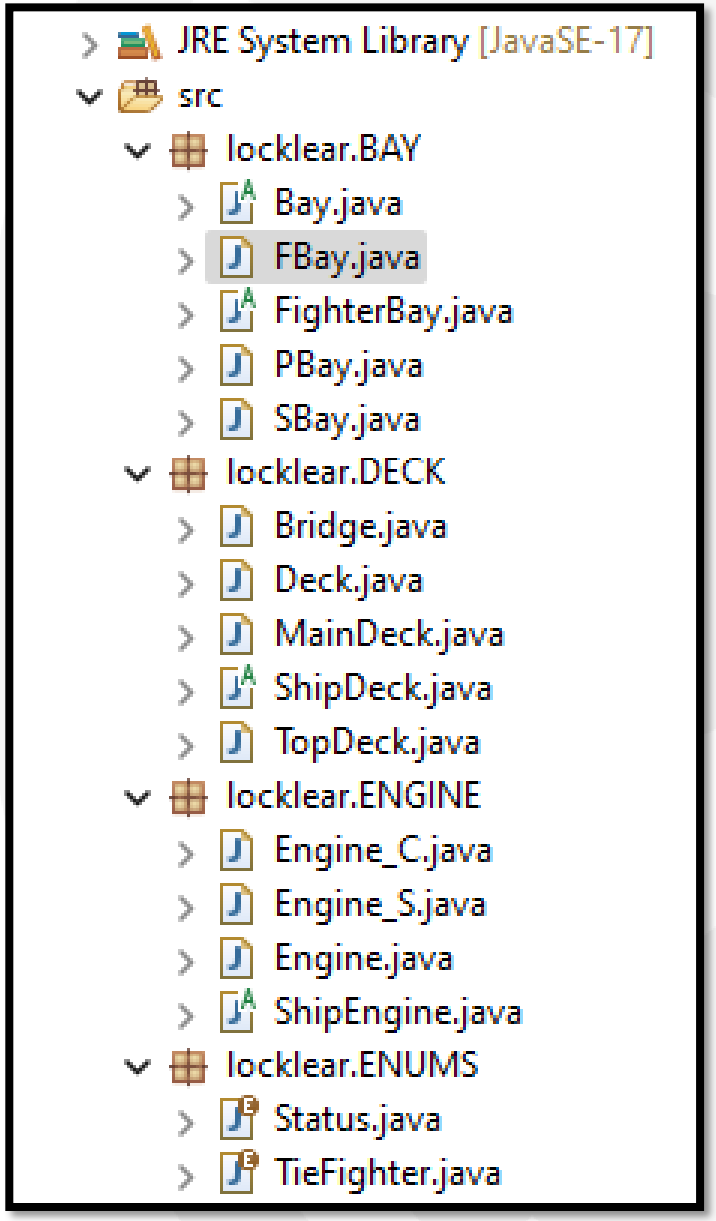




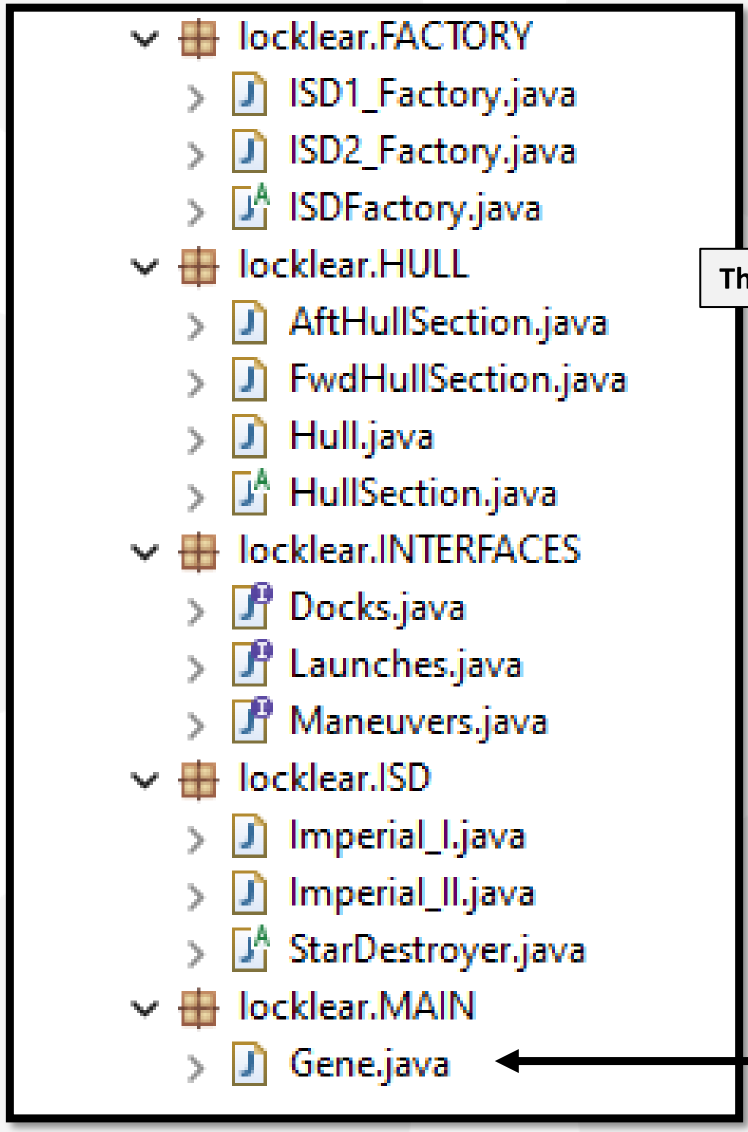




Program Structure



```
> JRE System Library [JavaSE-17]
v src
  v locklear.BAY
    > Bay.java
    > FBay.java
    > FighterBay.java
    > PBay.java
    > SBay.java
  v locklear.DECK
    > Bridge.java
    > Deck.java
    > MainDeck.java
    > ShipDeck.java
    > TopDeck.java
  v locklear.ENGINE
    > Engine_C.java
    > Engine_S.java
    > Engine.java
    > ShipEngine.java
  v locklear.ENUMS
    > Status.java
    > TieFighter.java
```

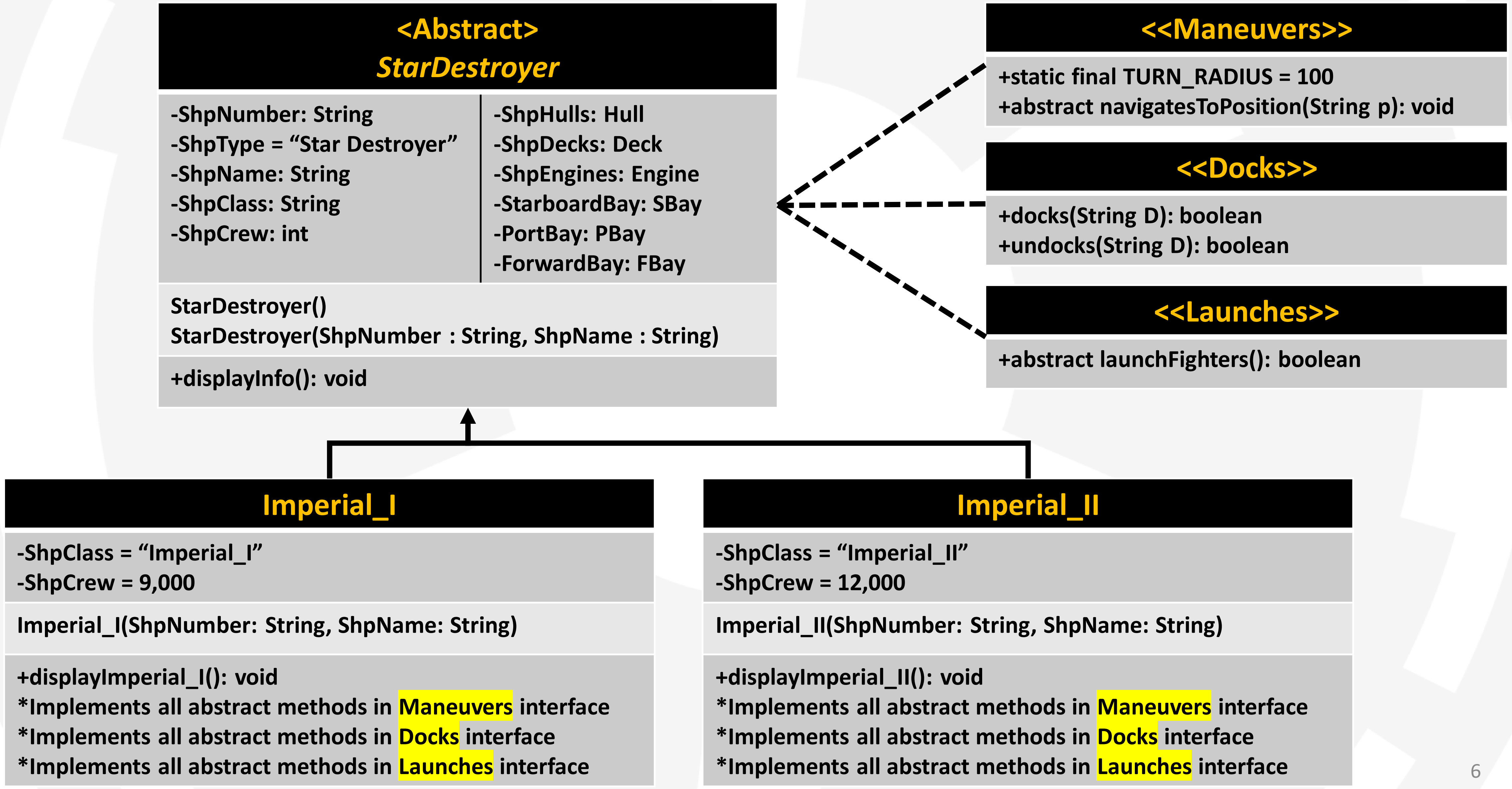


```
  v locklear.FACTORY
    > ISD1_Factory.java
    > ISD2_Factory.java
    > ISDFactory.java
  v locklear.HULL
    > AftHullSection.java
    > FwdHullSection.java
    > Hull.java
    > HullSection.java
  v locklear.INTERFACES
    > Docks.java
    > Launches.java
    > Maneuvers.java
  v locklear.ISD
    > Imperial_I.java
    > Imperial_II.java
    > StarDestroyer.java
  v locklear.MAIN
    > Gene.java
```

This is the structure I recommend…you may choose your own

This Class contains your validation (main method) of the aspects of your solution…you have the discretion to provide this validation as you believe it should be….The output from the main method in this class is how I will evaluate your solution.

StarDestroyer Class



<Abstract>
StarDestroyer

-ShpNumber: String
-ShpType = “Star Destroyer”
-ShpName: String
-ShpClass: String
-ShpCrew: int

-ShpHulls: Hull
-ShpDecks: Deck
-ShpEngines: Engine
-StarboardBay: SBay
-PortBay: PBay
-ForwardBay: FBay

StarDestroyer()
StarDestroyer(ShpNumber : String, ShpName : String)

+displayInfo(): void

<<Maneuvers>>

+static final TURN_RADIUS = 100
+abstract navigatesToPosition(String p): void

<<Docks>>

+docks(String D): boolean
+undocks(String D): boolean

<<Launches>>

+abstract launchFighters(): boolean

Imperial_I

-ShpClass = “Imperial_I”
-ShpCrew = 9,000

Imperial_I(ShpNumber: String, ShpName: String)

+displayImperial_I(): void
*Implements all abstract methods in Maneuvers interface
*Implements all abstract methods in Docks interface
*Implements all abstract methods in Launches interface

Imperial_II

-ShpClass = “Imperial_II”
-ShpCrew = 12,000

Imperial_II(ShpNumber: String, ShpName: String)

+displayImperial_II(): void
*Implements all abstract methods in Maneuvers interface
*Implements all abstract methods in Docks interface
*Implements all abstract methods in Launches interface

Component(HULL)

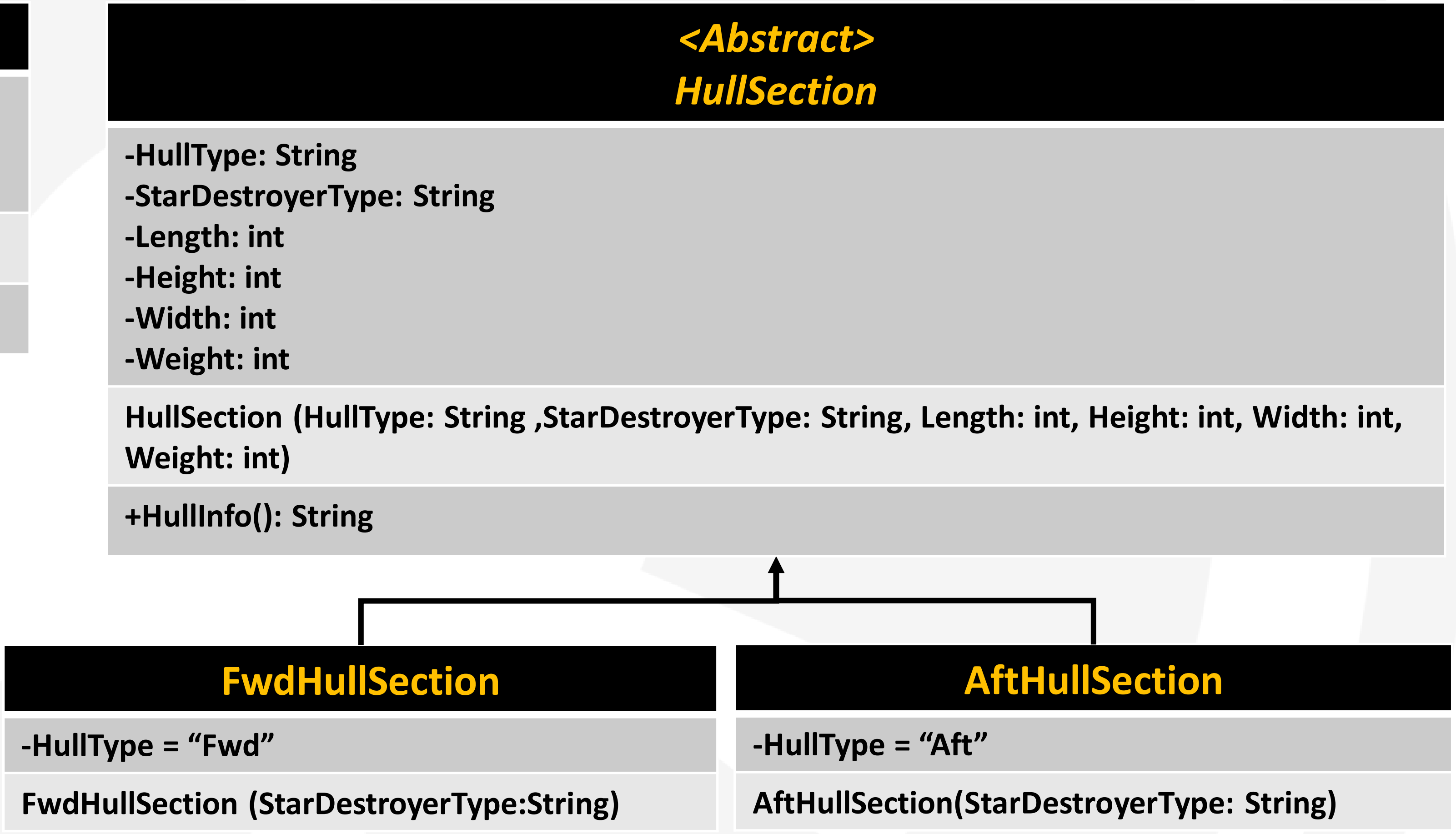
Hull

- IDNumber: String
- SectionFWD: FwdHullSection
- SectionAFT: AftHullSection

Hull(IDNumber: String, StarDestroyerType: String)

+displayHullSpecs(): void

Attribute	Hull	Imperial_I	Imperial_II
Length	Fwd	900 meters	900 meters
	Aft	700 meters	900 meters
Height	Fwd	100 meters	100 meters
	Aft	200 meters	275 meters
Width	Fwd	75 meters	200 meters
	Aft	325 meters	300 meters
Weight	Fwd	350 ISO tons	400 ISO tons
	Aft	550 ISO tons	700 ISO tons



<Abstract>
HullSection

-HullType: String
-StarDestroyerType: String
-Length: int
-Height: int
-Width: int
-Weight: int

HullSection (HullType: String ,StarDestroyerType: String, Length: int, Height: int, Width: int, Weight: int)

+HullInfo(): String

FwdHullSection

-HullType = “Fwd”

FwdHullSection (StarDestroyerType:String)

AftHullSection

-HullType = “Aft”

AftHullSection(StarDestroyerType: String)

Component Class(DECK)

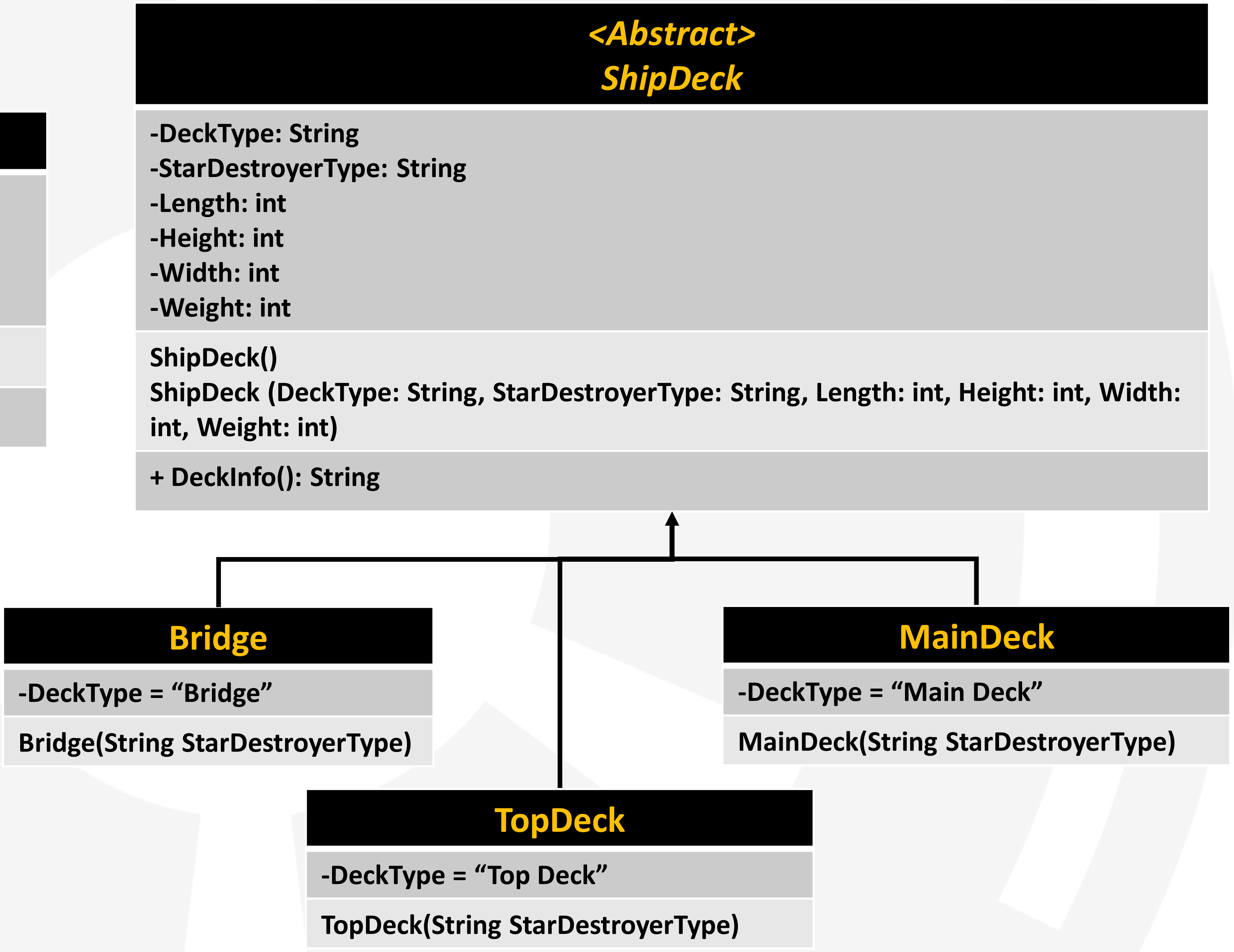
Deck

- IDNumber: String
- Deck_Bridge: Bridge
- Deck_Top: TopDeck
- Deck_Main: MainDeck

Deck(IDNumber: String, String StarDestroyerType)

+ displayDeckSpecs(): void

Attribute	Deck	Imperial_I	Imperial_II
Length	Bridge	150 meters	200 meters
	TopDeck	300 meters	350 meters
	MainDeck	600 meters	700 meters
Height	Bridge	20 meters	22 meters
	TopDeck	30 meters	34 meters
	MainDeck	50 meters	56 meters
Width	Bridge	100 meters	150 meters
	TopDeck	250 meters	300 meters
	MainDeck	400 meters	450 meters
Weight	Bridge	75 ISO tons	100 ISO tons
	TopDeck	125 ISO tons	150 ISO tons
	MainDeck	250 ISO tons	300 ISO tons



<Abstract>
ShipDeck

-DeckType: String
-StarDestroyerType: String
-Length: int
-Height: int
-Width: int
-Weight: int

ShipDeck()
ShipDeck (DeckType: String, StarDestroyerType: String, Length: int, Height: int, Width: int, Weight: int)

+ DeckInfo(): String

Bridge

-DeckType = "Bridge"

Bridge(String StarDestroyerType)

MainDeck

-DeckType = "Main Deck"

MainDeck(String StarDestroyerType)

TopDeck

-DeckType = "Top Deck"

TopDeck(String StarDestroyerType)

Component(ENGINE)

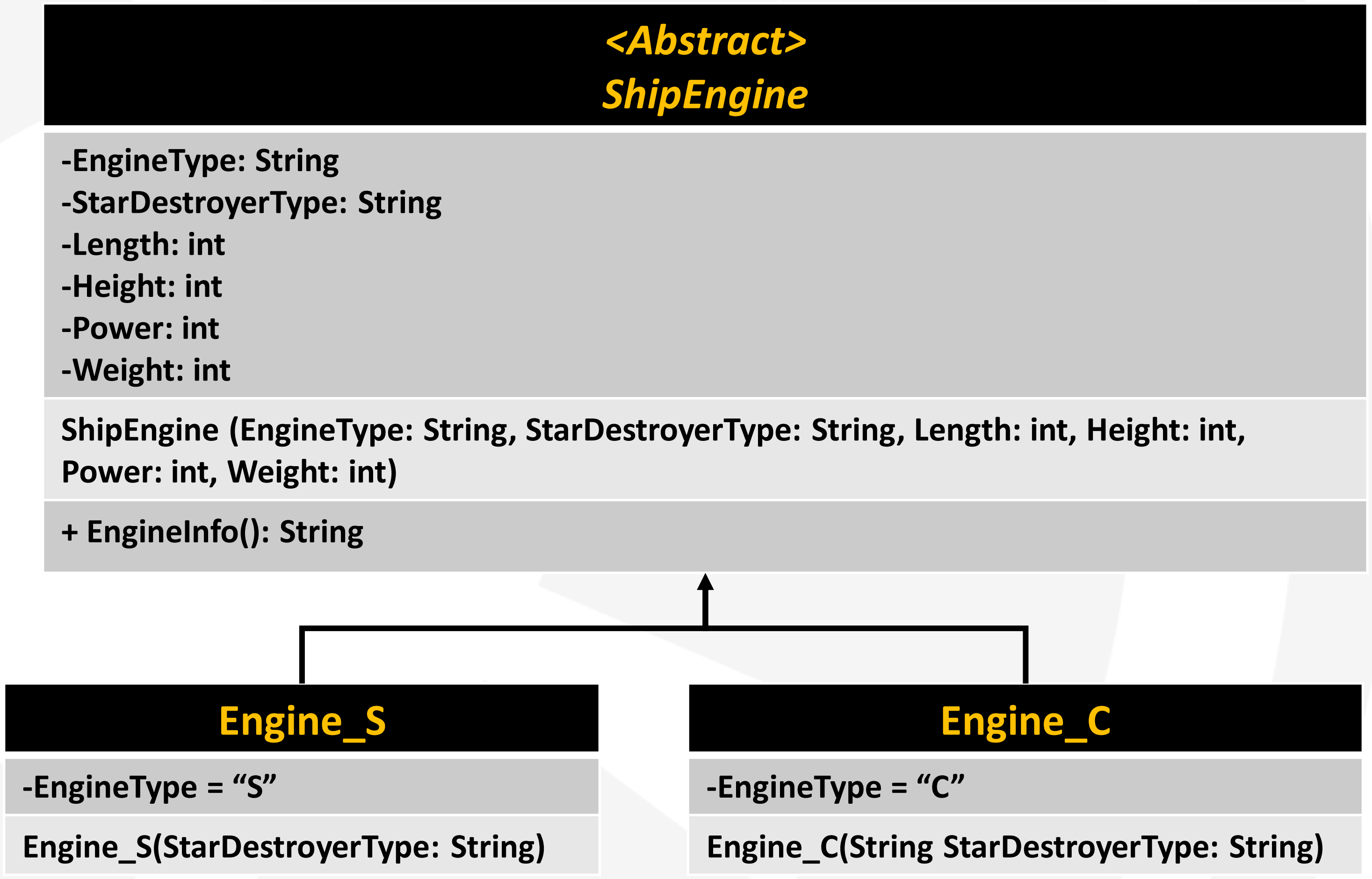
Engine

-IDNumber: String
-Engines: ShipEngine[ ]

Engine(IDNumber: String, StarDestroyerType: String)

+ displayEngineSpecs(): void

Attribute	Deck	Imperial_I	Imperial_II
Length	Engine_S	100 meters	115 meters
	Engine_C	225 meters	275 meters
Height	Engine_S	50 meters	60 meters
	Engine_C	75 meters	80 meters
Power	Engine_S	8 x 10^6 ISOs	9 x 10^6 ISOs
	Engine_C	15 x 10^6 ISOs	17 x 10^6 ISOs
Weight	Engine_S	100 ISO tons	100 ISO tons
	Engine_C	125 ISO tons	125 ISO tons



<Abstract>
ShipEngine

-EngineType: String
-StarDestroyerType: String
-Length: int
-Height: int
-Power: int
-Weight: int

ShipEngine (EngineType: String, StarDestroyerType: String, Length: int, Height: int, Power: int, Weight: int)

+ EngineInfo(): String

Engine_S

-EngineType = “S”

Engine_S(StarDestroyerType: String)

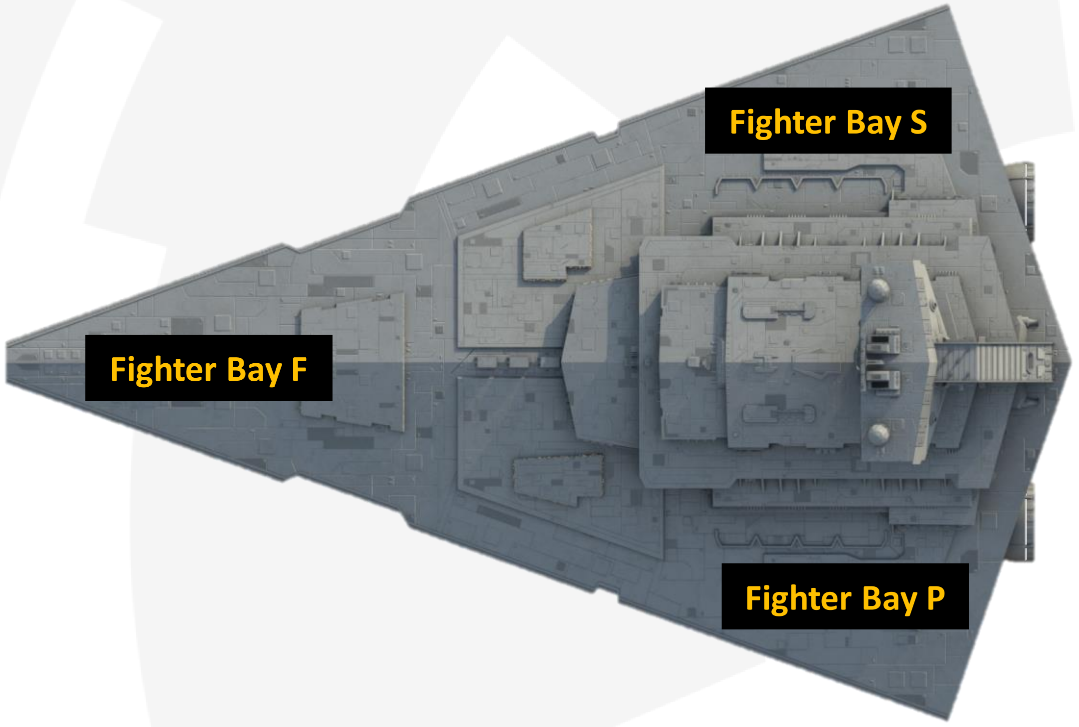
Engine_C

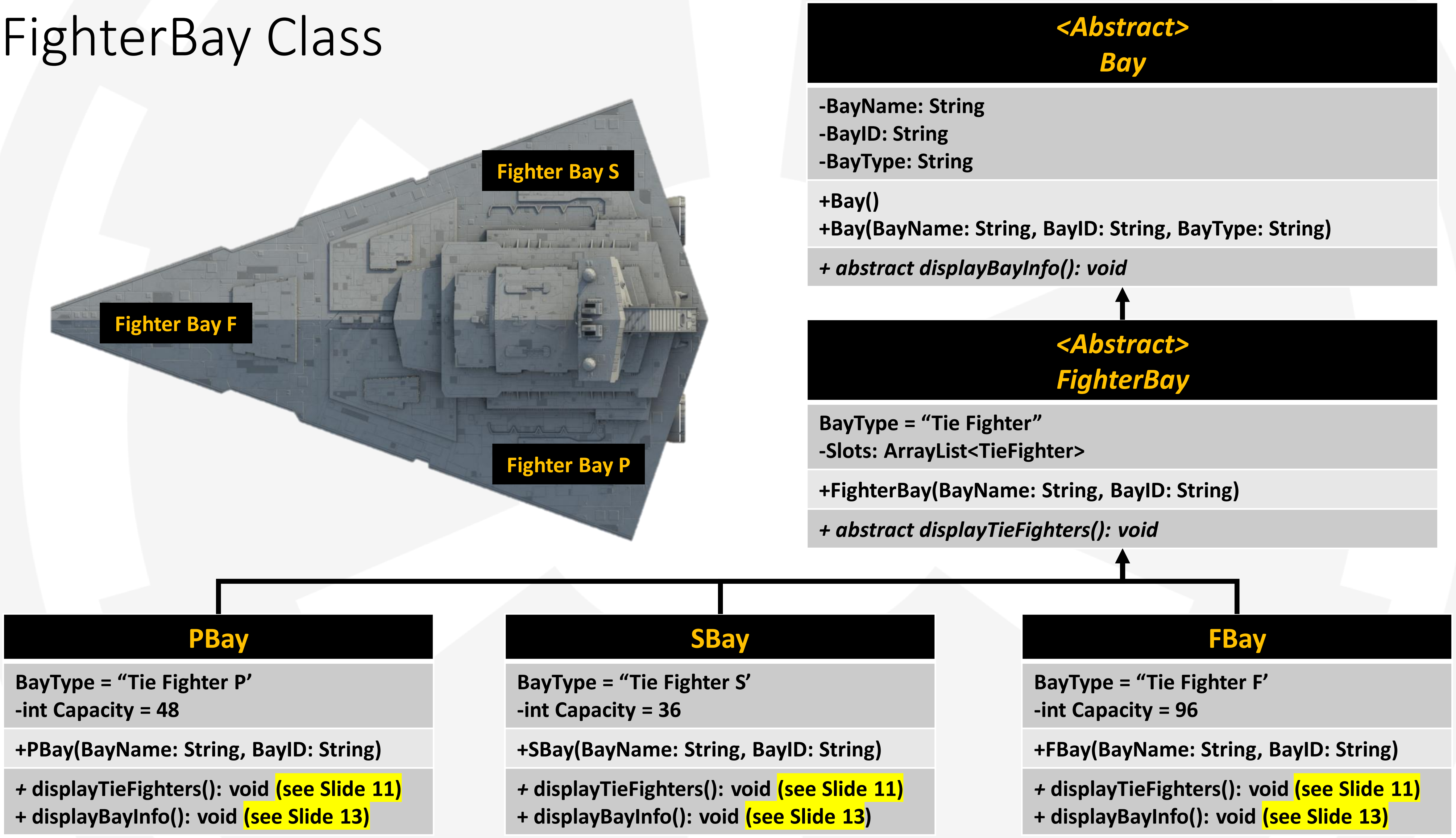
-EngineType = “C”

Engine_C(String StarDestroyerType: String)

Each Imperial Star Destroyer has three engines two of type S and one of type C

FighterBay Class





<Abstract>
Bay

-BayName: String
-BayID: String
-BayType: String

+Bay()
+Bay(BayName: String, BayID: String, BayType: String)

+ abstract displayBayInfo(): void

<Abstract>
FighterBay

BayType = "Tie Fighter"
-Slots: ArrayList<TieFighter>

+FighterBay(BayName: String, BayID: String)

+ abstract displayTieFighters(): void

PBay

BayType = "Tie Fighter P'
-int Capacity = 48

+PBay(BayName: String, BayID: String)

+ displayTieFighters(): void (see Slide 11)
+ displayBayInfo(): void (see Slide 13)

SBay

BayType = "Tie Fighter S'
-int Capacity = 36

+SBay(BayName: String, BayID: String)

+ displayTieFighters(): void (see Slide 11)
+ displayBayInfo(): void (see Slide 13)

FBay

BayType = "Tie Fighter F'
-int Capacity = 96

+FBay(BayName: String, BayID: String)

+ displayTieFighters(): void (see Slide 11)
+ displayBayInfo(): void (see Slide 13)

TieFighter Enumeration

<ENUM> TieFighter

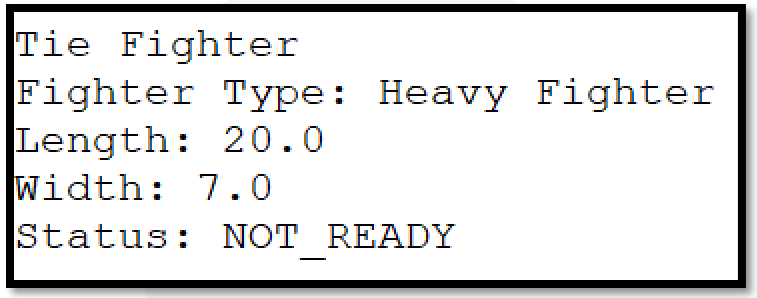
TieFighter_S(“Standard Fighter”,15,5,”NOT_READY”)
TieFighter_H(“Heavy Fighter”,20,7,”NOT_READY”)
TieFighter_I(“Stealth Fighter”,10,6,”NOT_READY”)

-TF_type: String -TF_length: double -TF_width: double -TF_status: Status
+ready(): void +display(): void

<ENUM> Status
READY, READYING, NOT_READY

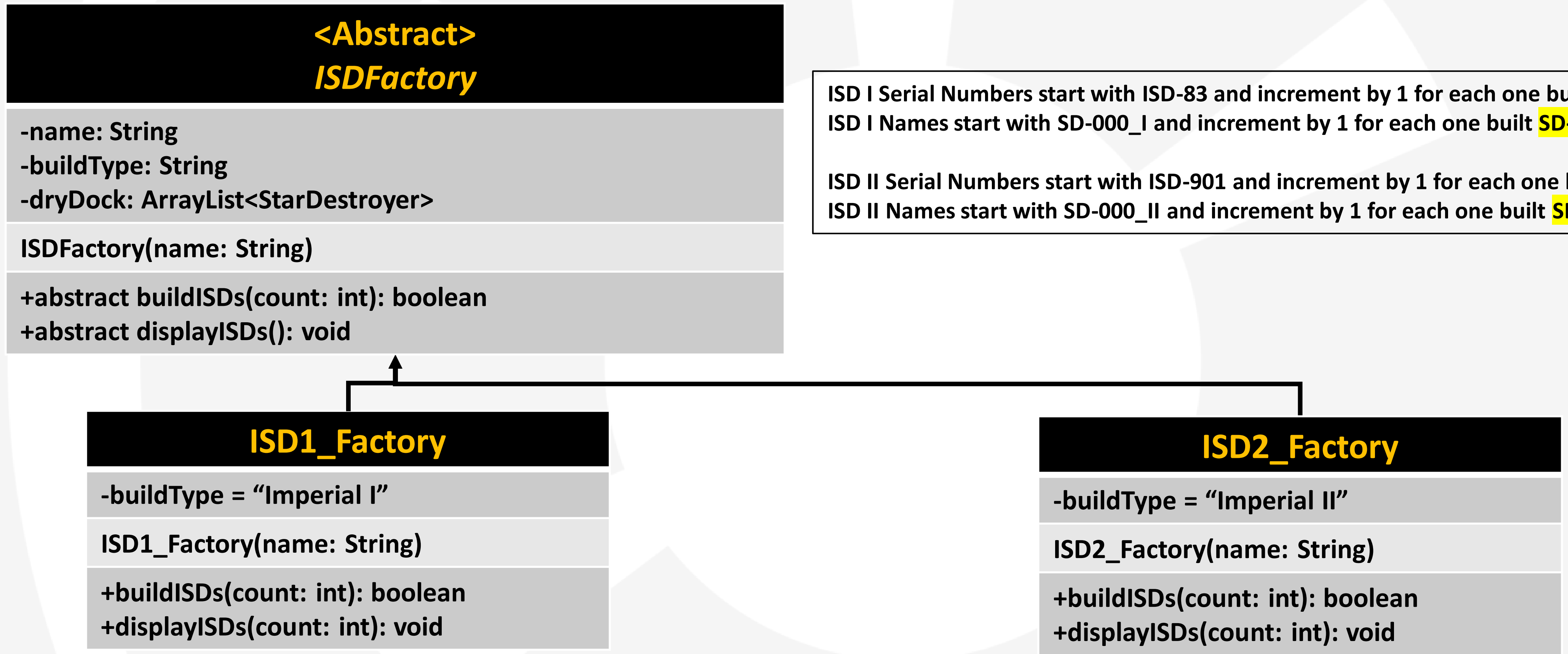
TieFighter_S are stored in the Forward Fighter Bay
TieFighter_H are stored in the Starboard Fighter Bay
TieFighter_I are stored in the Port Fighter Bay

Method Specification Chart	
ready	changes the Status of the TieFighter to **READY**
display	prints to console information about the TieFighter as shown here



```
Tie Fighter
Fighter Type: Heavy Fighter
Length: 20.0
Width: 7.0
Status: NOT_READY
```

ISDFactory Class

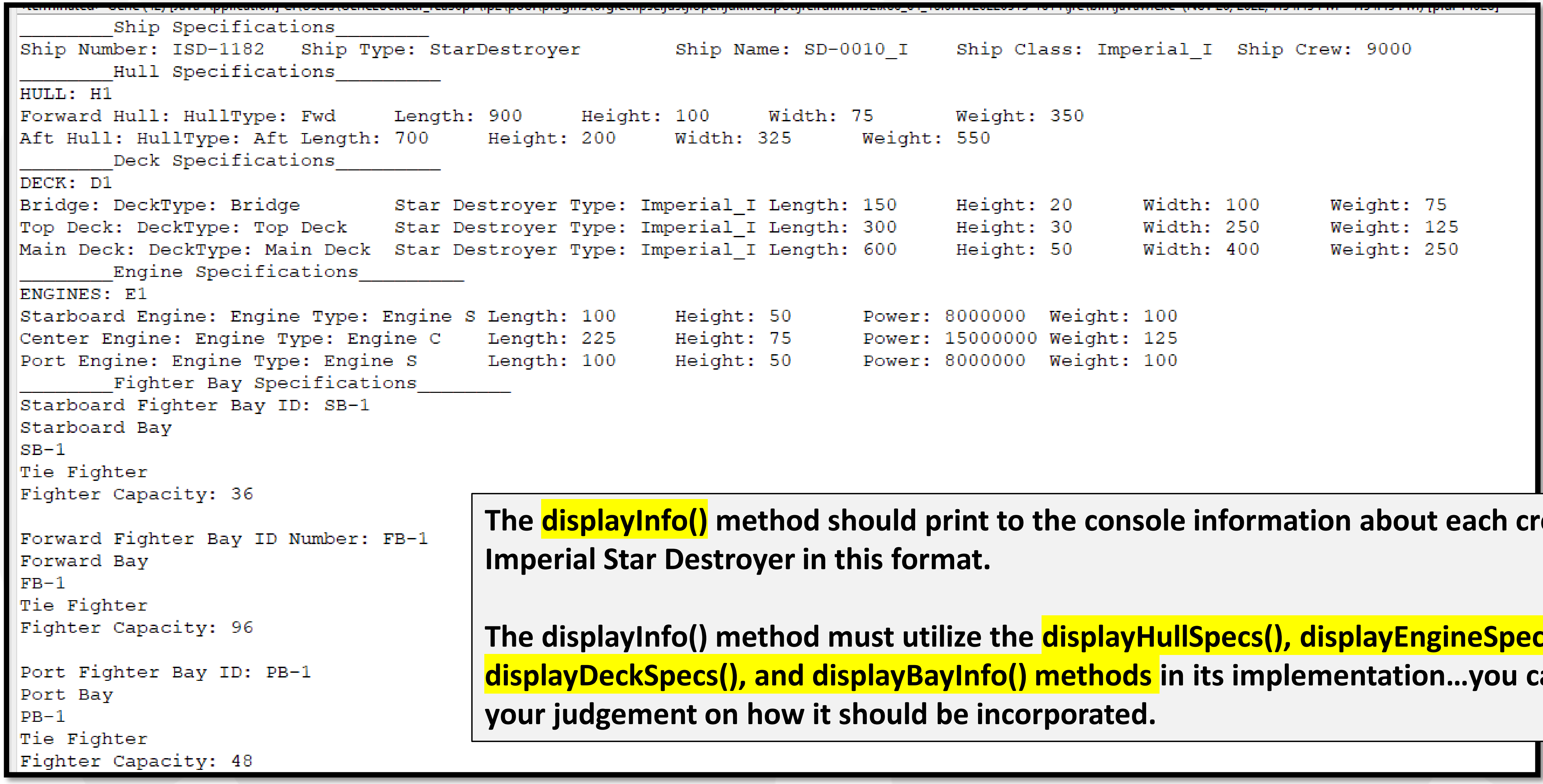

<Abstract>
ISDFactory
-name: String
-buildType: String
-dryDock: ArrayList<StarDestroyer>
ISDFactory(name: String)
+abstract buildISDs(count: int): boolean
+abstract displayISDs(): void
ISD1_Factory
-buildType = "Imperial I"
ISD1_Factory(name: String)
+buildISDs(count: int): boolean
+displayISDs(count: int): void
ISD2_Factory
-buildType = "Imperial II"
ISD2_Factory(name: String)
+buildISDs(count: int): boolean
+displayISDs(count: int): void

ISD I Serial Numbers start with ISD-83 and increment by 1 for each one built ISD-83, ISD-84, etc.
ISD I Names start with SD-000_I and increment by 1 for each one built SD-001_I, SD-002_I, etc.

ISD II Serial Numbers start with ISD-901 and increment by 1 for each one built ISD-901, ISD-902, etc.
ISD II Names start with SD-000_II and increment by 1 for each one built SD-001_II, SD-002_II, etc.

Individual Component ID Numbers begin with their first letter and then 1 or 2 depending on the ISD type.
EXAMPLE:
Imperial_I Hull ID Number would be H1
Imperial_II Engine ID Number would be E2

Console Output



```
________Ship Specifications________
Ship Number: ISD-1182   Ship Type: StarDestroyer        Ship Name: SD-0010_I    Ship Class: Imperial_I  Ship Crew: 9000
________Hull Specifications_________
HULL: H1
Forward Hull: HullType: Fwd     Length: 900     Height: 100     Width: 75       Weight: 350
Aft Hull: HullType: Aft Length: 700     Height: 200     Width: 325      Weight: 550
________Deck Specifications_________
DECK: D1
Bridge: DeckType: Bridge        Star Destroyer Type: Imperial_I Length: 150     Height: 20      Width: 100      Weight: 75
Top Deck: DeckType: Top Deck    Star Destroyer Type: Imperial_I Length: 300     Height: 30      Width: 250      Weight: 125
Main Deck: DeckType: Main Deck  Star Destroyer Type: Imperial_I Length: 600     Height: 50      Width: 400      Weight: 250
________Engine Specifications_________
ENGINES: E1
Starboard Engine: Engine Type: Engine S Length: 100     Height: 50      Power: 8000000  Weight: 100
Center Engine: Engine Type: Engine C    Length: 225     Height: 75      Power: 15000000 Weight: 125
Port Engine: Engine Type: Engine S      Length: 100     Height: 50      Power: 8000000  Weight: 100
________Fighter Bay Specifications________
Starboard Fighter Bay ID: SB-1
Starboard Bay
SB-1
Tie Fighter
Fighter Capacity: 36

Forward Fighter Bay ID Number: FB-1
Forward Bay
FB-1
Tie Fighter
Fighter Capacity: 96

Port Fighter Bay ID: PB-1
Port Bay
PB-1
Tie Fighter
Fighter Capacity: 48
```

The displayInfo() method should print to the console information about each created Imperial Star Destroyer in this format.

The displayInfo() method must utilize the displayHullSpecs(), displayEngineSpecs(), displayDeckSpecs(), and displayBayInfo() methods in its implementation…you can use your judgement on how it should be incorporated.