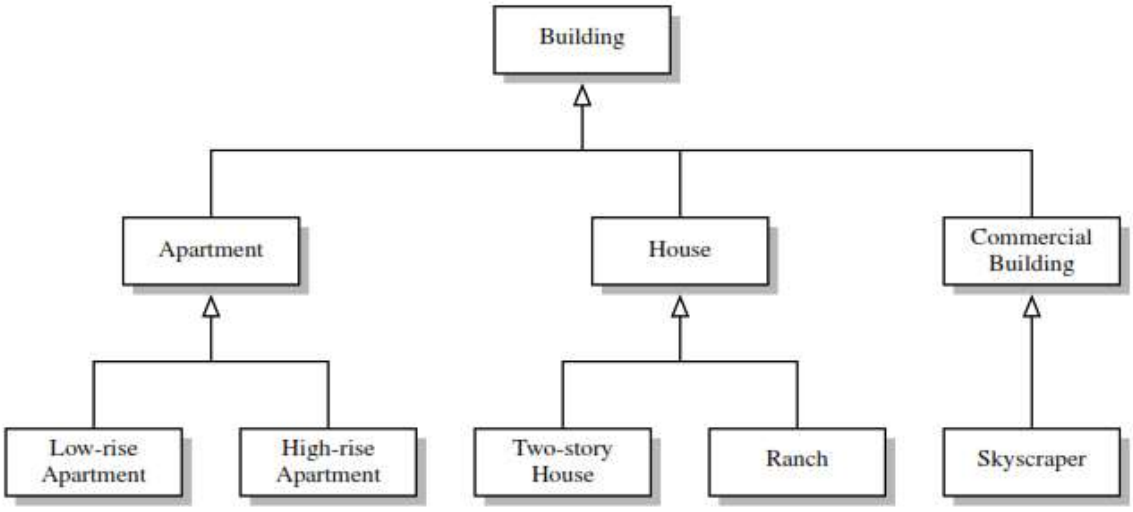
یادداشت‌های درسی (۳)

برای آنکه مردمان در کار خویش خرّم و خندان باشند سه شرط لازم است: نخست آنکه باید آن کار مناسب و در خور توان و استعداد ایشان باشد؛ دوم آنکه در آن کار افراط نکنند و بیش از حد بدان نپردازند؛ و سوم آنکه باید نوعی احساس موفقیت در دلشان نسبت به آن کار احساس شود.

جان راسکین (۱۹۰۰-۱۸۱۹)

ارث‌بری (Inheritance)

- گاهی طراح و نویسنده یک نرم‌افزار، نمی‌تواند تنها با تعریف یک کلاس، آنچه را که می‌خواهد به انجام برساند؛ یعنی ناچار است بیشتر از یک کلاس تعریف کند. یک راه برای کوچک‌تر کردن و ساده‌تر کردن برنامه در چنین مواقعی، این است که طراح، به گونه‌ای کلاس‌های برنامه را به هم مرتبط کند که صفات و توابعی در یک کلاس تعریف شوند و در کلاسی دیگر، آن صفات و توابع، عیناً یا با تغییراتی، مورد استفاده قرار گیرند؛ یا به بیان معروف در دنیای برنامه‌نویسی شیءگرا، یک کلاس، صفات و توابع کلاس دیگر را به ارث ببرد.
- **ارث‌بری**، مرسوم‌ترین راه برای تبیین ارتباط بین کلاس‌های مختلف یک برنامه شیءگرا است. رابطه ارث‌بری بین دو کلاس، وقتی معنا پیدا خواهد کرد که اشیاء یک کلاس، خود مثال‌هایی از اشیاء کلاس دیگر باشند؛ یعنی رابطه بین اشیاء دو کلاس، یک رابطه **is-a** باشد.
- رابطه ارث‌بری بین کلاس‌ها را می‌توان با یک ساختار سلسله – مراتبی نمایش داد. در چنین ساختاری، کلاسی که در پایین‌تر باشد، صفات و توابع تعریف شده در کلاسی که بالای آن قرار گرفته باشد را به ارث خواهد برد. به عنوان مثال، رابطه ساختمان‌ها (از نظر معماری آنها) را می‌توان با یک ساختار سلسله – مراتبی نمایش داد. طبق این ساختار، هر آسمان‌خراش **یک** ساختمان تجاری **است** و هر ساختمان تجاری **یک** ساختمان **است**؛ یا هر آپارتمان کم‌طبقه **یک** آپارتمان **است** و هر آپارتمان **یک** ساختمان **است**.



- به کلاسی که کلاس دیگر از آن ارث می‌برد، base class (کلاس پایه) یا superclass (اَبَرکلاس) یا parent class (کلاس والد) گفته می‌شود. و به کلاسی که از کلاس دیگر ارث می‌برد، derived class (کلاس مشتق) یا subclass (زیرکلاس) یا child class (کلاس فرزند) گفته می‌شود.
- اشیاء کلاس مشتق، همان صفات اشیاء کلاس پایه را دارند و می‌توانند همان رفتارهای اشیاء کلاس پایه را نیز داشته باشند. اما می‌توان در کلاس مشتق، تغییراتی در رفتارهای اشیاء ایجاد کرد؛ یا می‌توان علاوه بر صفات و رفتارهای تعریف شده برای اشیاء کلاس پایه، صفات و رفتارهای دیگری را نیز برای اشیاء تعریف کرد.

مثال ۱: ارث‌بری یگانه

در این مثال، یک کلاس پایه تعریف شده است به نام FootBall که اشیاء عضو آن، کشورهای عضو FIFA هستند؛ و یک کلاس مشتق تعریف شده است به نام WorldChampions که اشیاء عضو آن، کشورهایی هستند که عضو FIFA باشند و در دوره یا دوره‌هایی قهرمان جام جهانی فوتبال شده باشند. واضح است که هر قهرمان جام جهانی، **یک** کشور عضو FIFA **است**.

```
1.  class FootBall:
2.    def __init__(self, country, division, no_of_times):
3.      self.country = country
4.      self.division = division
5.      self.no_of_times = no_of_times
6.    def fifa(self):
7.      print(f"{self.country} national football team is placed in '{self.division}' FIFA division")
8.  class WorldChampions(FootBall):
9.    def world_championship(self):
10.     print(f"{self.country} national football team is {self.no_of_times} times world champions")
11. def main():
12.   germany = WorldChampions("Germany", "UEFA", 4)
13.   germany.fifa()
14.   germany.world_championship()
15. if __name__ == "__main__":
16.   main()
```

- از خط ۱ تا خط ۷ ، کلاس پایه FootBall تعریف شده است. از خط ۲ تا خط ۵ کلاس پایه، تابع سازنده ()__init__ تعریف شده است و سه صفت (attribute) برای اشیاء عضو کلاس (تیم‌های فوتبال) در نظر گرفته شده است: country ، division و no_of_times . در خطوط ۶ و ۷ کلاس پایه، تابع ()fifa تعریف شده است.
- از خط ۸ تا خط ۱۰ ، کلاس مشتق WorldChampions تعریف شده است. این کلاس، علاوه بر ارث بردن توابع و صفات عضو کلاس پایه، شامل تابعی ویژه اشیاء عضو خود به نام ()world_championship است.
- در خط ۱۲ ، شیءای به نام germany از کلاس مشتق WorldChampions ایجاد شده است. چون تابع سازنده ()__init__ صریحاً در کلاس مشتق تعریف نشده، تابع سازنده ()__init__ که صریحاً در کلاس پایه تعریف شده، برای ایجاد شیء germany فرا خوانی شده است.
- در خطوط ۱۳ و ۱۴ ، از طریق این شیء، تابع ()fifa (که در کلاس پایه تعریف شده است) و سپس تابع ()world_championship (که در کلاس مشتق تعریف شده است) فرا خوانی شده‌اند.

مثال ۲: استفاده از تابع ()super

اگر برنامه‌نویس بخواهد صفاتی مخصوص اشیاء کلاس مشتق تعریف کند، لازم است تابع سازنده ()__init__ را صریحاً در کلاس مشتق تعریف کند. اما برای استفاده از صفات تعریف شده در کلاس پایه، لازم است که آن تابع سازنده ()__init__ که صریحاً در کلاس پایه تعریف شده است، نیز فرا خوانی شود. می‌توان با استفاده از یک تابع نهفته (built-in) در زبان پایتون به نام ()super ، تابع سازنده در کلاس پایه را در بدنه تابع سازنده کلاس مشتق، فراخوانی کرد (این تابع، بدون نیاز به ذکر صریح نام کلاس پایه، توابع آن را فراخوانی می‌کند).

```
1.  class Country:
2.      def __init__(self, country_name):
3.          self.country_name = country_name
4.      def country_details(self):
5.          print(f"Happiest Country in the world is {self.country_name}")
6.  class HappiestCountry(Country):
7.      def __init__(self, country_name, continent):
8.          super().__init__(country_name)
9.          self.continent = continent
10.     def happy_country_details(self):
11.         print(f"Happiest Country in the world is {self.country_name} and is in {self.continent} ")
12. def main():
13.     finland = HappiestCountry("Finland", "Europe")
14.     finland.happy_country_details()
15. if __name__ == "__main__":
16.     main()
```

- از خط ۱ تا خط ۵ ، کلاس پایه Country تعریف شده است. در خطوط ۲ و ۳ ، تابع سازنده __init__() تعریف شده است و یک صفت برای اشیاء عضو کلاس پایه (کشورها) در نظر گرفته شده است: country_name . در خطوط ۴ و ۵ ، تابع country_details() تعریف شده است.
- از خط ۶ تا خط ۱۱ ، کلاس مشتق HappiestCountry تعریف شده است (واضح است که شادترین کشور، **یک** کشور **است**). تابع سازنده __init__() صریحاً در این کلاس با دو پارامتر country_name و continent تعریف شده است. از آنجا که این تابع سازنده، جایگزین تابع سازنده کلاس پایه به حساب می‌آید، برای دسترسی و مقداردهی اولیه به صفت country_name ، با استفاده از تابع super() ، تابع سازنده کلاس پایه Country فراخوانی شده است. در خط ۹ نیز، صفت جدید continent مقداردهی شده است.
- در خط ۱۳ ، شیءای به نام finland از کلاس مشتق HappiestCountry ایجاد شده است. و در خط ۱۴ ، از طریق این شیء، تابع happy_country_details() (که در کلاس مشتق تعریف شده است) فرا خوانی شده است.

مثال ۳: method overriding

گاهی برنامه‌نویس لازم می‌داند که تعریف یک تابع در کلاس پایه را در کلاس مشتق تغییر دهد. به این کار method overriding گفته می‌شود. با این کار، دو تابع با امضای یکسان (با نام و ترتیب و تعداد پارامترهای یکسان) در کلاس پایه و کلاس مشتق وجود خواهد داشت. می‌توان با استفاده از تابع super() ، به آن نسخه از تابعی که در کلاس پایه تعریف شده (اما در کلاس مشتق override شده است) دسترسی داشت.

- از خط ۱ تا خط ۶ ، کلاس پایه Book تعریف شده است. در خطوط ۲ و ۳ و ۴ ، تابع سازنده __init__() تعریف شده است و دو صفت برای اشیاء عضو کلاس (کتاب‌ها) در نظر گرفته شده است: author (نویسنده) و title (عنوان). در خطوط ۵ و ۶ کلاس پایه، تابع book_info() تعریف شده است.

```
1.  class Book:
2.      def __init__(self, author, title):
3.          self.author = author
4.          self.title = title
5.      def book_info(self):
6.          print(f"{self.title} is authored by {self.author}")
7.  class Fiction(Book):
8.      def __init__(self, author, title, publisher):
9.          super().__init__(author, title)
10.         self.publisher = publisher
11.     def book_info(self):
12.         print(f"{self.title} is authored by {self.author} and published by {self.publisher}")
13.     def invoke_base_class_method(self):
14.         super().book_info()
15. def main():
16.     print("Derived Class")
17.     silva_book = Fiction("Daniel Silva", "Prince of Fire", "Berkley")
18.     silva_book.book_info()
19.     silva_book.invoke_base_class_method()
20.     print("----------------------------------")
21.     print("Base Class")
22.     reacher_book = Book("Lee Child", "One Shot")
23.     reacher_book.book_info()
24. if __name__ == "__main__":
25.     main()
```

- از خط ۷ تا خط ۱۴ ، کلاس مشتق Fiction تعریف شده است. تابع سازنده __init__() صریحاً در این کلاس تعریف شده است و در بدنه آن، با استفاده از تابع super() ، تابع سازنده کلاس پایه Book فراخوانی شده است.
- در خطوط ۱۱ و ۱۲ ، تابع book_info(self) که در کلاس پایه تعریف شده، override شده است. در خطوط ۱۳ و ۱۴ ، تابعی به نام invoke_base_class_method(self) تعریف شده است و با استفاده از تابع super() در بدنه آن، آن نسخه از تابع book_info(self) که در کلاس پایه تعریف شده، فرا خوانی شده است.
- در خط ۱۷ ، شیءای به نام silva_book از کلاس مشتق Fiction ایجاد شده است. و در خطوط ۱۸ و ۱۹ ، از طریق این شیء، آن نسخه از تابع book_info() که در کلاس مشتق تعریف شده است و تابع invoke_base_class_method() فرا خوانی شده است.
- در خط ۲۲ ، شیءای به نام reacher_book از کلاس پایه Book ایجاد شده است. و در خط ۲۳ ، از طریق این شیء، آن نسخه از تابع book_info() که در کلاس پایه تعریف شده، فرا خوانی شده است.

*****

تمرین : برنامهای شیءگرا بنویسید که مختصات نقطه مرکز یک دایره و مقدار شعاع دایره و همچنین مختصات یک نقطه را بگیرد و تعیین کند که آن نقطه، درون دایره قرار دارد یا روی محیط دایره یا خارج از دایره.