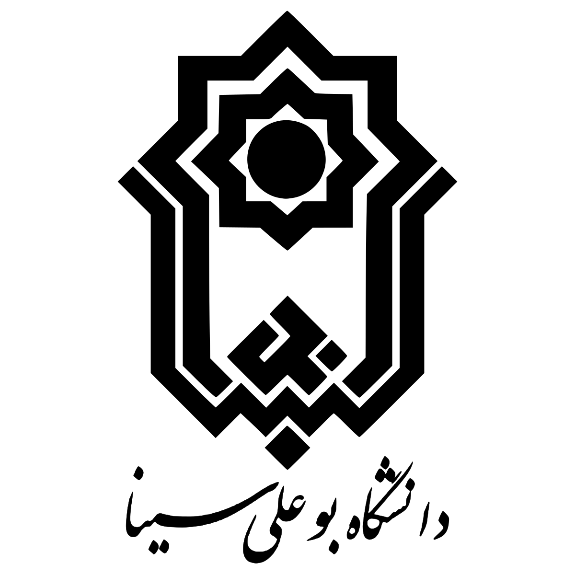
بسمه تعالی



عنوان:

**«گزارش کار فاز سوم پروژه پایانی درس برنامه سازی پیشرفته»**

**«Sewing Game»**

استاد:

**سرکار خانم مهندس بطحائیان**

دانشجو:

**نسترن منصوری**

**9612358041**

ترم تحصیلی:

**4001**

فهرست

[هدف: 3](#_Toc90530907)

[توضیحات کلی کد در این فاز: 3](#_Toc90530908)

[کلاس Manto: 3](#_Toc90530909)

[کانستراکتور: 4](#_Toc90530910)

[توابع set\_button و set\_brooch: 5](#_Toc90530911)

[کلاس Pants: 6](#_Toc90530912)

[Pants.h: 6](#_Toc90530913)

[Pants.cpp: 7](#_Toc90530914)

[کانستراکتور: 7](#_Toc90530915)

[تابع set\_pants-size: 7](#_Toc90530916)

[تابع set\_belt: 8](#_Toc90530917)

[کلاس Design: 9](#_Toc90530918)

[Design.h: 9](#_Toc90530919)

[Design.cpp: 10](#_Toc90530920)

[کانستراکتور: 10](#_Toc90530921)

[تابع choose\_design: 10](#_Toc90530922)

[تابع set\_design: 11](#_Toc90530923)

[تابع get\_design: 11](#_Toc90530924)

[تابع save\_design: 12](#_Toc90530925)

[تابع get\_save\_design: 12](#_Toc90530926)

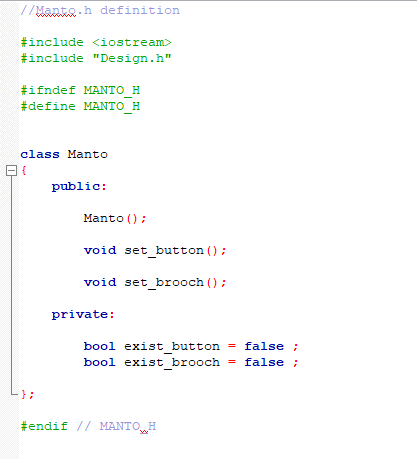
# هدف:

هدف از پیاده سازی این فاز استفاده از وکتورها و پیاده سازی استثناها می باشد.

# توضیحات کلی کد در این فاز:

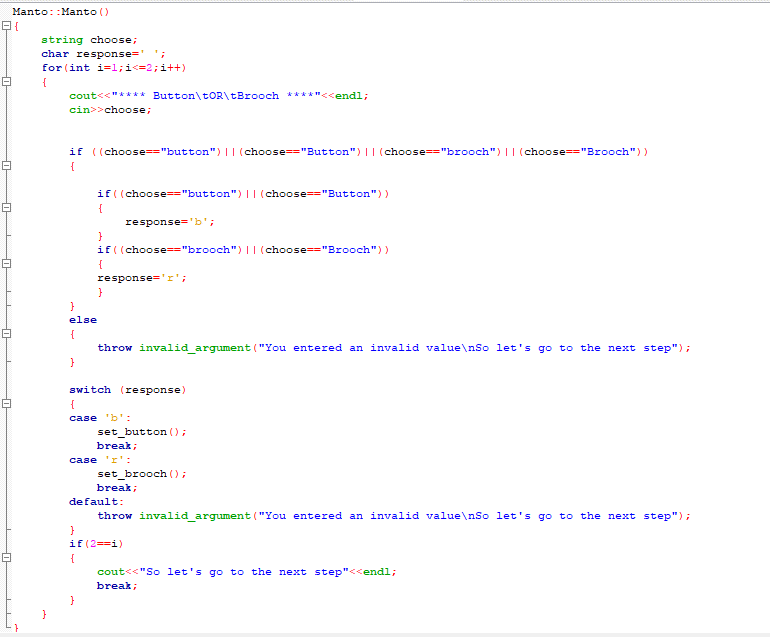
در این فاز علاوه بر تکمیل ارث بری های موجود سعی شده است تا از وکتور و استثناها هم استفاده شود. به این منظور کلاس Manto و کلاس Design تغییر پیدا کرده و کلاس جدید تحت عنوان Pants تعریف شده است. در ادامه تغییرات موجود در دو کلاس Manto و Design و پیاده سازی کلاس Pants به طور کامل توضیح داده شده است.

# کلاس Manto:



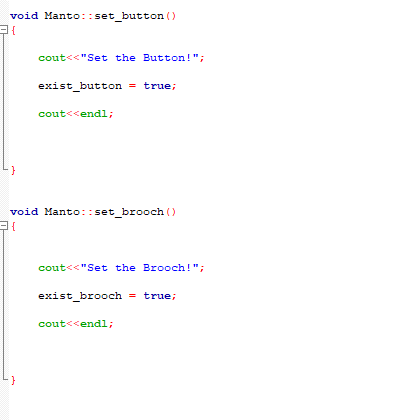
در header این کلاس تغییری ایجاد نشده است اما پیاده سازی کانستراکتور و دو تابع set\_button و set\_brooch آن به طور کامل تغییر یافته است. ابتدا به توضیح کانستراکتور می پردازیم.

## کانستراکتور:



همانطور که در تصویر مشخص است در قسمت اول کد یک متغیر از نوع string به نام choose و یک متغیر از نوع char به نام response تعریف شده است. هنگامی که یک شی از این کلاس ایجاد می شود و کانستراکتو رآن فراخوانی می شود، ابتدا وارد یک حلقه می شویم. حلقه ای به طول 2؛به این منور که از کاربر پرسیده میشود که میخواهد دکه و یا سنجاق سینه داشته باشد یا خیر؟ حال کاربر باید مشخص کند که می خواهد دکمه یا سنجاق سینه داشته باشد یا خیر اگر بخواهد که باید button یا brooch را وارد کند و مقدار ورودی در choose ذخیره می شود. حال در این قسمت کامپایلر چک میکند که ورودی کاربر در صورت خواست دکمه یا سنجاق سینه درست باشد. اگر ورودی کاربر دکمه باشد مقدار 'b' در response ذخیره می شود واگر ورودی کاربر سنجاق سینه باشد مقدار 'r' در response ذخیره می شود. اگر ورودی کاربر هیچ یک از این مقادیر نبوده باشد و درواقع choose به طور اشتباه پر شود، استثنایی پرتاب خواهد شد با این مضمون که "مقدار ورودی استباه است پس به مرحله بعد می رویم". اما اگر استثنا رخ ندهد و برنامه به صورت عادی عمل کند در قسمت بعدی با استفاده از switch…case مقدار موجود در response چک می شود تا مشخص شود که کدام یک از توابع set\_button و set\_brooch فراخوانی شود. در این قسمت هم اگر مقدار response هیچ یک از مقادیر مورد نظر نباشد باز هم استثنا پرتاب خواهد شد.

## توابع set\_button و set\_brooch:

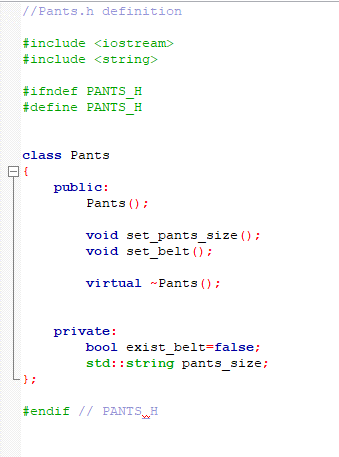


همانطور که در تصویر مشخص است این دو تابع نسبت به فاز قبلی دچار تغییر شده اند. تغییرات آن ها به این صورت است که با فراخوانی هریک پیغای تحت عنوان "سنجاق سینه ست شد" و "دکمه ست شد" داده می شود و متغیر های exist\_brooch و exist\_button از false به true تغییر خواهند یافت.

# کلاس Pants:

این کلاس، کلاس شلوار است. این کلاس هم ماند=ند سایر کلاس های موجوددر پروژه از دو قسمت header و .cpp تشکیل شده است که در ادامه توضیح داده می شود.

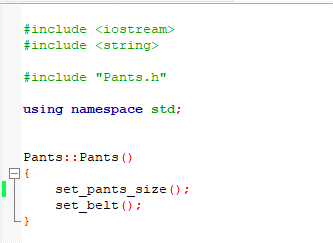
# Pants.h:



همانطور که مشخص است این کلاس دو متغیر خصوصی دارد به نام های exist\_belt که از نوع Boolean بوده و برای مشخص کردن حضور یا عدم حضور کمربند می باشد و pants\_size که از نوع رشته بوده و برای ذخیره سازی بلند یا کوتاه بودن شلوار است. همچنین دو تابع به غیر از کانستراکتور آن وجود دارد به نام set\_pants\_size و set\_belt.

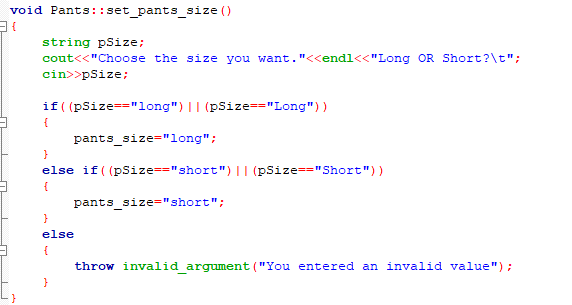
# Pants.cpp:

## کانستراکتور:



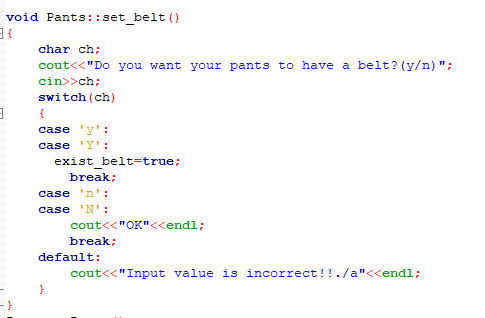
در کانستراکتور این کلاس تنها تابع set\_belt و تابع set\_pants-size فراخوانی می شود.

## تابع set\_pants-size:



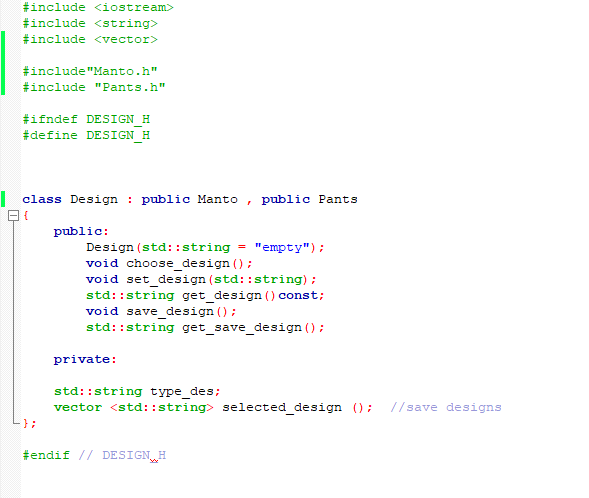
این تابع متغیری دارد از نوع رشته که سایز شلوار در ان نگه میدارد. از کاربر خواسته میشود که سایز مورد نظر خود را انتخاب و وارد کند. سپس مقدار ورودی در pSize ذخیره می شود. بعد چک می شود که مقدار آن long بوده یا short که اگر هریک از اینها باشد و درست نوشته شده باشد در متغیر خصوصی pants\_size ذخیره می شود. اگر مقدار ورودی اشتباه باشد استثنا رخ می دهد.

## تابع set\_belt:



این تابع هم متغیر از نوع کاراکتر دارد که از کاربر پرسیده می شودکه آیا میخواهد شلوارش کمربند داشته باشد یا خیر؟ سپس مقدار ورودی توسط کاربر در متغیر کاراکتری ch ذخیره می شودو بعد با استفاده از switch…case چک می شود که اگر مقدار ورودی 'y' بود مقدار متغیر خصوصی exist\_belt ،true می شود در غیر اینصورت مقدار پیش فرض آن که false بود باقی میماند و یک پیغام "OK" چاپ خواهد شد. اگر مقدار ورودی هیچ یک از مقادیر موردنظر نباشد هم یک استثنا رخ می دهد.

# کلاس Design:

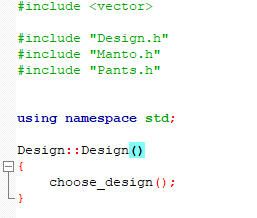


# Design.h:

در قسمت header این کلاس مشخص است که از دو کلاس Manto و Pants ارث میبرد. دو متغیر خصوصی دارد به نام های type\_des از نوع رشته برای ذخیره نوع طراحی کاربر و selected\_design از نوع وکتور برای ذخیره طراحی های انتخاب شده در طول بازی. همچنین به غیر از کانستراکتور توابع دیگری هم وجود دارد در قسمت public این کلاس که در بخش .cpp توضیح داده خواهد شد.

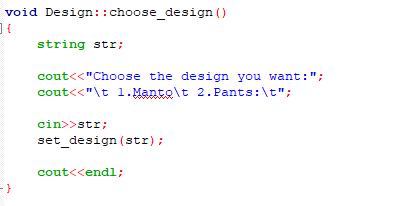
# Design.cpp:

کانستراکتور:



در کانستراکتور این کلاس فقط تابع choose\_design فراخوانی می شود.

## تابع choose\_design:



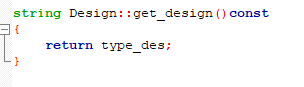
این تابع متغیری از نوع رشته دارد تا مقدار منتخب و ورودی کاربر از طراحی های موجود را ذخیره کند؛ سپس با این متغیر تابع set\_design فراخوانی می شود.

## تابع set\_design:



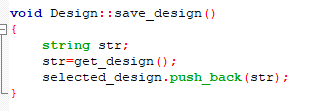
در این تابع در ابتدا مقدار ورودی طراحی توسط کاربر در متغیر خصوصی کلاس یعنی type\_des ذخیره می شود. پس چک می شود که مقدار ورودی manto است یا pants؟ اگر هیچ یک از دو مورد نبود استثنا رخ می دهد. در نهایت تابع save\_design برای ذخیره سازی طراحی ها فراخوانی می شود.

## تابع get\_design:



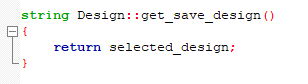
طراحی منتخب توسط کاربر را برمیگرداند.

## تابع save\_design:



این تابع طراحی های انتخابی توسط کاربر در طول بازی را در وکتور selected\_design ذخیره می کند.

## تابع get\_save\_design:



این تابع مقادیر موجود در وکتور طراحی ها را برمیگرداند.