الف

به توضیح هر یک از موارد ذکر شده به ترتیب می پردازیم:

- Polymorphism یا چند ریختی : ویژگی از زبان java است که به ما این اجازه را میدهد که یک child object بتواند در جای parent object خود قرار بگیرد و به توان از روی parent object بعضی از behaviour های child object را صدا زد و behaviour به ان صورتی که در child تعریف شده است انجام شود.
- Substitution : در واقع یکی از ویژگی های زبان java و اکثر زبان oop است که در آنها این اجازه به برنامه نویس داده میشود که یک reference به یک object بتواند در فضایی که مختص به parent یا ancestor های آن است (در واقع متغیری که type آنها مربوط به سطح های بالاتری از آن object است)ذخیره بشود پس در واقع می توان این را هم گفت که این ویژگی به method ها این اجازه را میدهد که ورودی از نوع parent داشته باشند و هنگام صدا زدن ما child انها را به انها پاس بدهیم.
- Abstract class : گاها در هنگام طراحی یک شیا که به صورت parent است ما از جزئیات دقیق یک method یا یک field در child object باخبر نیستیم یا به صورت کلی نمی خواهیم class یی که در حال نوشتن کد ان هستم را بتوان از ان نمونه ای ساخت و به نوعی می خواهیم این class فقط یک parent برای class های دیگر باشد و تعدادی behaviour برای انها تعریف کند و از طرفی ضرورتی ایجاد کند که خود انها چند behaviour دیگر داشته باشند که خود تعریف میکنند ؛ به این نوع class ها abstract class میگویند
- Interface: در مهندسی نرم افزار گاها ما در شرایطی که به diamond problem معروف است که نیاز داریم یک class از دو یا چند class دیگر ارث بری کند برای حل این مشکل در زبان های oop مفهومی به نام interface ارایه شد و در زبان های دیگری مفهومی multi inheritance یا prototypal inheritance اجرا شد. در این روش ما تمام behaviour های ان class را که باید داشته باشد تعریف میکنیم اما اینکه دقیقا چه کدی باید اجرا شود را تعیین نمی کنیم به این صورت وقتی ان class ارث بری را انجام میدهد دیگر خودش دقیقا کدی را که باید برای این behaviour اجرا شود را تعیین می کند و در واقع نقش interface مانند یک ضرورت دهنده است که بودن یک سری behaviour ها را بر ان class ضروری میکند و به نوعی همان interface را برای class مشخص میکند.

ب

1. False : in inheritance we extend some other class to inherit its behaviours, if so what is the point of being able to inherit the class own behaviours
2. True: In java in order for a subclass to exist, the parent class must exist to, so the compiler will insert a no argument super call if there is no mention of that in the subclass constructor and if there is no constructor with zero arguments then the program gets run time error
3. True: a subclass inherits all members and constructors aren't regarded as members in java.
4. True: in overriding you can make it more accessible but there is no restriction for method overloading
5. False: the whole point of private methods is that no one can see them so the method in the subclass has no way of knowing that there is a private method with that specifications and so no overriding will take place.
6. True: As mentioned in the last question interfaces were introduced to solve the 'diamond problem' so it makes sense that classes must be able to implement more than 1 interface.

ج

با توجه به اینکه در صورت سوال پاسخ کامل خواسته شده است ابتدا یک تعریف برای هر کدام از این ویژگی ها مطرح می شود سپس انها را با هم مقایسه میکنم

- Method overloading : یک ویژگی در زبان های oop است به این صورت که می توان چندین method را با یک اسم مشخص داشت که هر کدام از آنها در واقع تعداد و یا نوع ورودی هایی که میگیرند با هم متفاوت است و بعد نسبت به آنها رفتار های متفاوت یا مشابهی باهم دارند.
- Method overriding : یک ویژگی دیگر از زبان های oop است به این صورت که یک تابع که دقیقا اسم و نوع و تعداد ورودی های آن با یک تابع دیگر که در parent یا ancestor های آن است ؛ برابر است و ما می خواهیم اگر بروی subclass ان method را صدا زدیم کد دیگری اجرا شود نسبت به آنکه روی subclass ان method را صدا بزنیم پس در این حالت method overriding را انجام میدهیم

در مورد تفاوت های آنها می توان به موارد زیر اشاره کرد :

- Override کردن در واقع مربوط به ارث بری است و اگر ارث بری نداشته باشیم دیگر هیچ override کردنی صورت نمی پذیرد اما overload کردن هم بین parent و child هم بین method های یک class میتواند باشد
- از طرفی overloading در واقع compile time polymorphism را پیاده میکند و overriding در واقع runtime polymorphism را پیاده میکند.
- همانطور که در بالا هم گفته شد در overriding هر دو یک signature دارند اما در overloading باید signature ها متفاوت باشد.