سوال اول:

الف : برای درک بهتر این سه مفهوم در oop آنها را با ترتیب مشخصی تعریف می کنیم :

- State : در پارادایم OOP مهمترین مفهوم در اشیا state آنها است زیرا همین مجموعه ی state است که state کلی برنامه را تعریف می کند ؛ حال state به معنی مجموعه یا حالت فعلی مقادیر property های object است و مفهوم ان به این شکل است که object ساخته شده اکنون در چه حالی است و این state این امکان را دارد که تغییر پیدا کند یا حتی نکند
- Behaviour : همانطور که در بالا هم اشاره شد که state یک object مهم ترین مفهوم است و همچنین این state این امکان را دارد که تغییر پیدا کند ؛ حال ما ابزاری نیاز داریم که این state را تغییر بدیم و یا به صورت کلی رفتاری برای ان object تعریف کنیم به این شکل که اگر در object های دیگه ان را صدا زدیم ان object چه رفتاری از خودش نشان می دهد یا اگر خواستیم state آن را عوض کنیم ان object چه رفتاری از خودش نشان می دهد مجموعه ی این رفتار ها behaviour ان object است
- Identity : این مفهوم در oop به این موضوع اشاره دارد که هر object بالاخره باید به نوعی در حافظه ذخیره شود و این محل ذخیره سازی به نوعی identity ان object است و به این دلیل است که دو نمونه از یک class حتی با state کاملا برابر با هم برابر نیستند زیرا در واقع دو نمونه ی متفاوت و دو فضای متفاوت در حافظه را به خود اختصاص داده اند

ب: نحوه ی تعامل اشیا در زبان های مختلف می تواند شیوه های مختلف داشته باشد در زبان java بدون در نظر گرفتن مطالبی که هنوز در کلاس تدریس بحثی از آنها نشده است (مانند inheritance یا polymorphism و . . .) می توان گفت که یک شیء با یک شیء با یک اشیا دیگر از طریق درست کردن یک نمونه از آن و فعال کردن یک behaviour از آن object با آن در ارتباط است ؛ البته این مطلب را هم می توان اضافه کرد که می توان توابع و متغیر ها را به صورت static هم تعریف کرد به این صورت که دیگر لزوما نیازی به ساختن یک نمونه از آن class نیست تا بتوان از آن behaviour دسترسی پیدا کرد.

ج : هر کدام از این مفاهیم را که بعضی به صورت کلی در هر زبان oop هستند و بعضی هم در تمام زبان ها نیستند را به شکل زیر تعریف می کنیم :

- Overloading : این مفهوم در تمام زبان ها به یک شیوه انجام نشده است اما به نوعی میتوان ادعا کرد خود مفهوم به نوعی در هر زبانی است و حتی شاید لزوما مربوط به oop نیست. در اساسی ترین حالت این کلمه به این فلسفه اشاره دارد که برنامه نویس با توجه به شرایط صدا سازی یک تابع (مانند ورودی ها یا static صدا کردن تابع یا نه و . . .) بتواند قطعه کدی را که اجرا می شود ؛ در نتیجه behaviour آن را ؛ تغییر دهد در زبان java به این صورت انجام میشود که اسم تابع ثابت است اما با تغییر نوع ورودی ها (نه لزوما اسم شان) یک تابع overload میشود.
- Casting : این مفهوم در تمام زبان های مدرن موجود است و لزوما ربطی به oop ندارد به این معنا است که type یک متغیر را عوض کرد ؛ به این صورت که ان که ان قسمتی از زبان را که وظیفه ی اجرای کد را در اختیار دارد ان صفر و یک هایی را که برای ان متغیر ذخیره کرده است به یک نوع دیگر (در صورت امکان ان) معنی کند .
- Modularization : این مفهوم نیز بازهم لزوما به oop اشاره ندارد اما در واقع oop یکی از هدف های درست شدنش برای درست تر انجام دادن modularization بود و در زبان های oop از جمله java به راحتی این مفهوم را پیاده کرد. این مفهوم به این موضوع اشاره دارد که یک سوال بزرگ را به این سوال های کوچک تر بشکنیم و هر کدام از آن مشکل های کوچکتر را حل کنیم و بعد هر کدام از این جواب ها را به نوعی پیاده سازی کنیم که دیگر هیچ وابستگی به اجزای دیگر سوال نداشته باشد و اگر در سوال دیگری بعد از شکستن سوال به سوال های کوچک تر دوباره همان سوال به وجود امد بتوان کدی را که در گذشته نوشته ایم با کمترین تغییرات ممکن (یا در بهترین حالت بدون هیچ تغییری) بتوان در حل سوال جدید استفاده کرد.
- Abstraction: این مفهوم کلی به این فلسفه در برنامه نویسی اشاره دارد که وقتی ما از یک ابزار (مانند یک تابع یا یک class یا ...) در کد خود استفاده می کنیم دیگر لازم نیست دقیقا از شیوه انجام یک عملیات در آن داخل آن خبر داشته باشیم بلکه فقط بر اساس behaviour ی که برای آن در documentation آن تعریف شده است بتوان از ان استفاده کرد . در این حالت دیگر برنامه نویسی که این ابزار را می سازد نگران نحوه ی دقیق پیاده سازی behavior آن ابزار است و دیگر برنامه نویس دیگری که از آن استفاده می کند صرفا از آن استفاده می کند و از نتیجه ی نهایی آن استفاده می کند .

د: در زبان های oop گاها به این حالت نیاز است که وقتی ما یک object از یک class میسازیم دیگر نتوانیم state آن را به هیچ وجه تغییر بدهیم و اگر بخواهیم یک state از آن را تغییر دهیم لازم باشد یک نمونه ی جدید با ان state مورد نظر ما درست کنیم .
یک نمونه ی خیلی معروف از آن در اکثر زبان های برنامه نویسی string است.
و خود این مفهوم میتواند به modularization بهتر و ساده تر کردن خود class ها کمک می کند.