به نام خدا

HashSet از HashMap برای ذخیره ی عناصر خود استفاده می کند در حالیکه TreeSet از TreeMap استفاده می کند.

HashSet ترتیب قرارگیری عناصر را حفظ نمی کند در حالیکه TreeSet عناصر را طبق مبنای داده شده(عملگر comparator) مرتب می کند و اگر مبنا داده نشود آن ها را بر اساس ترتیب صعودی عادیشان مرتب می کند.

TreeSet چون باید عناصر را بعد از هر اضافه کردن یا خذف کردن مرتب کند ،عملکرد ضعیف تری نسبت به HashSet ای دارد که عناصر خود را مرتب نمی کند.

Time complexity برای عملیات های اضافه کردن ، حذف کردن و بازیابی عنصر ، برای HashSet از مرتبه ی O(1) می باشد در حالیکه برای TreeSet این سه عملیات از مرتبه ی O(log(n)) می باشد(چون عناصر مرتب شده می باشند و بعد از Add یا Delete باید دوباره مرتب شوند.)

برای مقایسه ی عناصر و حذف عناصر مشابه احتمالی ، HashSet از دو تابع equals() و hashCode() استفاده می کند در حالیکه TreeSet از compare() و compareTo() بهره می گیرد.

در HashSet حداکثر یک عنصر Null می توانیم داشته باشیم در حالیکه در TreeSet امکان ذخیره ی Null وجود ندارد و در صورت اضافه کردن آن با NullPointerException مواجه می شویم.

HashSet در مقایسه با TreeSet حجم (فضا، memory ) کم تری اشغال می کند چون از HashMap بهره می گیرد و از طرفی TreeSet به دلیل نگه داشتن ترتیب عناصرخود بر طبق Comparator در کنار بهره گیری از TreeMap به حافظه یا حجم( memory ) بیشتری نیاز دارد.

در آخر ، اگر ترتیب قرارگیری عناصر برایمان اهمیتی نداشت از HashSet بهتر است استفاده کنیم چون عملیات در HashSet در مقایسه با TreeSet از time complexity کم تری برخوردار هستند و از طرفی HashSet فضا(memory) کمتری نسبت به TreeSet اشغال می کند.

اگر قصد مرتب کردن عناصر بر اساس یک مقایسه گر خاص را داشته باشیم استفاده از TreeSet مناسب این کار است.