سوال یک:

الف) هر یک از کلمات JDK, KRE, JVM به ترتیب مخفف Java Development Kit, Java Runtime enviroment, Java Virtual Machine هستند. ما در واقع می‌توانیم با نصب JDK به طور کامل JRE و JVM را نیز نصب کنیم. همانطور که از نام JDK مشخص است, JDK محیطی برای توسعه‌ی برنامه‌های جاوا است که دارای کامپایلر جاوا، خود زبان جاوا، دو بخش دیگر نام‌برده شده در این سوال و همچنین javadoc و jar است. ما برای ران کردن برنامه‌ی خود به محیط مناسبی نیاز داریم که آن محیط همان JRE است. این محیط دارای حداقل نیازها برای ران کردن یه برنامه‌ی نوشته‌شده به زبان جاوا است و شامل JVM، کلاس‌های هسته‌ای(مرکزی) و برخی فایل‌های دیگر است. در واقع JRE فقط برای ران کردن برنامه‌ها و نه توسعه دادن آن‌ها است. اما JVM یکی از مهم‌ترین بخش‌ها است که با نصب هر یک از موارد بالا نصب میشود. JVM در واقع وظیفه‌ی execute کردن برنامه و درحقیقت خواندن خط به خط کدها را برعهده دارد و گاهی به آن Interpreter نیز گفته میشود.

ب) به طور خلاصه میتوان گفت که در برنامه‌نویسی ساخت‌یافته ما میتوانیم از توابع مختلف و روابط بین آن‌ها استفاده کنیم تا برنامه‌ای را در آن زبان توسعه دهیم در حالیکه در برنامه‌نویسی شیئ‌گرا ما میتوانیم از اشیا و روابط بین اشیا برای نوشتن برنامه‌ی خود استفاده کنیم. از دیگر تفاوت‌های این دو میتوان به این اشاره کرد که modify کردن در برنامه‌نویسی شیئ‌گرا کار نسبتا راحت‌تری تا انجام این کار در برنامه‌نویسی ساخت‌یافته است. یکی دیگر از تفاوت‌ها نیز که در واقع برآمده از همان استفاده از اشیا است، وجود access specifierها است که ما در برنامه‌نویسی شیئ‌گرا میتوانیم از آن‌ها برای تعیین کرد سطح دسترسی به متغیرها، متودها و کلاس‌ها استفاده کنیم. همین قابلیت موجب این میشود که اطلاعات در برنامه‌هایی که با زبان‌های شیئ‌گرا نوشته شده‌اند دارای امنیت بیشتری باشند زیرا میتوان در آن‌ها کنترل بسیار بیشتر و راحت‌تری روی ارتباط کاربر با برنامه داشت. ما در واقع در برنامه‌نویسی ساخت‌یافته مسئله‌ی خود را به توابع مختلفی که هر کدام به جدایی زیرمسئله‌های مسئله‌ی اصلی را حل می‌کنند می‌شکنیم ولی در برنامه‌نویسی شیئ‌گرا سعی بر این است که برنامه را به اشیا مختلف و متودهای آن‌ها تقسیم کنیم و از روابط آن‌ها استفاده کنیم.

ج) مراحل کامپایل و اجرای یک برنامه‌ی نوشته‌شده را میتوان به دو بخش تقسیم کرد. در مرحله‌ی اول کامپایلر فایل‌های با پسوند .java را می‌خواند و از هر فایل با این پسوند یک فایل مجزا با پسوند .class می‌سازد. باید دقت داشت که در انجام این پروسه و هنگام خواندن کد، کامپایلر آن‌ها را به ماشین‌کد یعنی زبان قابل فهم برای کامپیوتر تبدیل می‌کند. ولی پس از انجام این کار یعنی به وجود آوردن فایل‌های .class، این فایل‌ها مستقل از سیستم‌عامل هستند و میتوانند روی هر دستگاهی که دارای JVM باشد اجرا شوند. در ابتدای توضیح تفاوت جاوا با سی میتوان به این اشاره کرد که به دلیل اینکه C یک زبان compiler و جاوا یک زبان Interpreter است، سرعت اجرای برنامه‌ها در زبان C سریع‌تر از جاوا است. ابتدا کد C وارد پردازنده‌ها میشود و precessor file ها به کد اصلی ما attach میشوند. Processor fileها همان هدرفایل‌هایی هستند که ما در برنامه‌ی خود Include کرده‌ایم. در نتیجه پس از این مرحله ما کدی با اندازه‌ی بیشتر (expanded code) داریم. پس از این کامپایلر کد ما را به زبان assembly تبدیل میکند و در مرحله‌ی بعد assembler آن را به object code تبدیل میکند. سپس

linker کد حاصل را به libraryهای مختلفی که در واقع بهشی از سورس‌کد ما نیستند ولی به اجرای برنامه کمک میکنند لینک میکند و در آخر loader کد ما را در Ram لود میکند و برنامه ران میشود.